Advanced Software Testing Volume 1

高级软件测试 卷1

高级软件测试分析师

Rex Black 著

刘 琴 周震漪
郑文强 马均飞 译



清华大学出版社

Advanced Software Testing Volume 1

内容简介

本书是目前市面上唯一完整的完全按照国际软件测试认证委员会（ISTQB）的测试人员认证高级大纲撰写的专业书籍系列之一。书本章节与高级大纲中高级软件测试分析师模块的标题、顺序和知识点相匹配。适宜ISTQB-ATA（高级软件测试分析师）的认证考前学习和有志于在测试高级分析技术领域有所建树的人员系统化的学习。

本书概要介绍了软件测试背景和基本概念，详细描述了实践性很强的基于风险分析的测试过程及活动；聚焦在高级软件测试分析师所要掌握的测试技术领域，重点介绍了软件测试分析与设计的技术和方法；结合ISO9126、ISO1044国际标准阐述了事件（缺陷）管理全生命周期模型及事件（缺陷）报告的撰写方法，并介绍了其他相关的国际标准和专业领域标准；此外也从自动化软件测试工具和测试团队管理等各个角度解释了获得ISTQB-ATA资质证书应该具备的各种条件。

本书每一章都提供与ISTQB高级大纲一致的、充足的认证考试模拟题，为准备参加ISTQB-ATA（高级软件测试分析师）资质认证考试的人员提供相应的基础性知识。每一章节的课堂练习都结合一个实际案例HELLOCARMS而来，并提供答案，具有较高的参考价值。

ISBN 978-7-302-24417-2
9 787302 244172 >
定价：39.00元

Advanced Software Testing Volume 1

高级软件测试 卷1

高级软件测试分析师

Rex Black 著

刘 琴 周震漪
郑文强 马均飞 译

清华大学出版社
北京

Simplified Chinese edition copyright © **2010** by **Rock Nook Inc.**
Original English language title from Proprietor's edition of the Work.

Original English language title: Advanced Software Testing, Volume 2 **by** Rex Black © **2010**
EISBN: 978-1-933952-19-2
All Rights Reserved.

Published by arrangement with the original publisher, **Rock Nook Inc.**

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).
本书中文简体翻译版由 **Rock Nook Inc.** 授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号 图字: 01-2009-7737 号

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。
版权所有,侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目 (CIP) 数据

高级软件测试·卷 1: 高级软件测试分析师/(美)布莱克(Black, R.)著;刘琴等译. --北京: 清华大学出版社, 2011.8
书名原文: Advanced Software Testing, Volume 1
ISBN 978-7-302-24417-2

Ⅰ. ①高… Ⅱ. ①布… ②刘… Ⅲ. ①软件—测试 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2010)第 260386 号

责任编辑: 龙啟铭
责任校对: 李建庄
责任印制: 何 芊

出版发行: 清华大学出版社 **地 址**: 北京清华大学学研大厦 A 座
http://www.tup.com.cn **邮 编**: 100084
社 总 机: 010-62770175 **邮 购**: 010-62786544
投稿与读者服务: 010-62795954, jsjjc@tup.tsinghua.edu.cn
质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者: 北京富博印刷有限公司
装 订 者: 北京市密云县京文制本装订厂
经 销: 全国新华书店
开 本: 185×260 **印 张**: 18.5 **字 数**: 462 千字
版 次: 2011 年 8 月第 1 版 **印 次**: 2011 年 8 月第 1 次印刷
印 数: 1～3000
定 价: 39.00 元

产品编号: 035688-01

译者序

作为国际软件测试认证委员会（ISTQB）大中华首席代表，我目睹了祖国软件产业日新月异的发展，以及从跟随到逐步开始影响国际软件测试技术潮流的趋势。在软件测试行业，ISTQB 在全世界的影响力日益壮大。截至 2011 年 5 月 ISTQB 成员国已经达到 48 个，全球证书持有者总数已经超过 16 万人。在测试市场逐步向亚太地区扩张的形势下，如何迅速提高英语为非母语的中国测试从业人员在测试领域的竞争力？最直接有效的途径是掌握国际通用的测试行业知识。ISTQB 为业界唯一公认的测试知识以及认证体系，其非营利及开放性不但吸纳融合了国际测试领域知识和经验的精华，并且确保了该体系的与时俱进。目前，译者正在参与的撰写中的国际软件与系统测试标准 ISO29119 也是与 ISTQB 知识体系完全兼容的。因此我们审慎地选择了向知名出版社推荐并引进基于 ISTQB 大纲撰写的基础级和高级系列教材。

回想 2006 年，在上海静安寺附近的一家小咖啡厅，本书的四位译者初次汇聚在一起，从翻译最基本的统一词汇开始，到目前大中华区分会（CSTQB）拥有 30 余位国内专家，目前已经完全引进 ISTQB 最新知识体系，中国证书持有者 3000 余人，有两位专家直接参与新版本 ISTQB 高级和专家级编写，并看到相关翻译教材的出版，我非常感慨。激励我们翻译本书的初衷就是为推动国内的软件测试行业的国际化、标准化尽一份微薄之力。虽然这本书是基于 ISTQB 高级大纲第一个模块编写的，许多案例还是从美国企业抽取的，与我国的国情和文化并不十分贴合。我也希望能够协助国内的业界精英，开发出基于本土案例的教材或者教学课件，更有效地推动，甚至参与 ISTQB 及相关知识体系的发展和创新。

本书的作者 Rex Black 先生是 ISTQB 的第三届主席，也是美国测试委员会的发起人之一，在测试界深具影响力和号召力。我在翻译本书期间一直和 Black 先生保持沟通，感谢他不厌其烦地释疑解难，并坦诚地接受个别译者对原文的疑问和建议。本书紧扣 ISTQB 高级认证大纲中高级软件测试分析师部分，而且包含了一些新的内容（例如基于风险分析的测试），适用于希望获得 ISTQB 高级资质认证证书的人群，包括担任任何与软件测试工作相关的角色的人员，如测试员、测试分析员、测试工程师、测试顾问、测试经理、用户验收测试员和软件开发人员等。同时也适合想对软件测试高级知识

有所了解的人，比如项目经理、质量经理、软件开发经理、业务分析师、IT主管和管理顾问等。

本书也是一本很实用的高校教材。目前本书作为ISTQB高级测试认证的官方认可的教材已成功在同济大学软件学院推广应用，本书的课件也得到学生的一致好评。同济大学软件学院正在结合该体系打造软件测试与质量保证实验室，提供国内第一个实践平台和实训课程。本书也已被CSTQB（Chinese Software Qualifications Testing Board，国际软件测试认证委员会大中华分会）指定为官方培训教材。虽然工程师的素养有赖于知识教育、实践能力和企业经验的融合，但通过对本书的学习，能够为高校及企业培养具有国际竞争力的软件测试工程师提供一个可参考的，与ISTQB对接的渠道。

在此我谨以ISTQB大中华首席代表和CSTQB理事长代表的名义，向共同担任翻译工作的周震漪、马均飞和郑文强表示最诚挚的谢意，同时也要感谢同济大学软件学院的师生花费宝贵时间，与我反复探讨书中的一些经典案例的译文。最后我心怀歉疚地感谢一直支持我的先生徐添和我三岁的女儿徐贝宁，没有你们的理解和支持，我难以想象在我本已繁忙的工作之余还能完成这本书的翻译。

由于水平和时间的限制，书中的翻译难免会出现错误，欢迎读者以及各界同仁不吝指正。

刘琴 教授 博士
同济大学软件学院常务副院长
ISTQB 中国首席代表
CSTQB 理事长代表
CCF 软件工程专业委员会委员
2011年5月

致谢

如果想把所有曾在我测试职业生涯中给过我帮助的人的名字一一列举出来，恐怕需要另外写一本书了。在此我仅向那些对我写本书影响最深的人士致谢。

首先，我想向我在美国软件测试认证委员会和国际软件测试认证委员会的同事致谢，尤其是和我在高级大纲的工作组中一起工作的人们。正是有了他们的努力，本书才得以成形。令我倍感荣幸的是，我不仅与本书的各位参与者互相分享和学习了彼此在测试方面的见解与观点，而且还蝉联了美国软件测试认证协会和国际软件测试协会的主席一职。在即将卸任主席职务之时，我为我们在软件测试领域所达到的成就感到非常的自豪，我将怀着同样的自豪感期待我们能在将来共同创造出更大的辉煌。同时我也期待本书能够恰当地表达我对我们在软件测试领域共同做出贡献的谢意与敬意。

接下来，我想向那些为本书提供素材的人们致谢。本书中包含的素材，以及高级软件测试分析师指导书、教程和高级软件测试分析师在线教程，都被他们反复地校对、推敲和完善。他们是 José Mata、Judy McKay、Jamie Mitchell、Paul Jorgensen 和 Pat Masters。

一旦本书中所需的素材创作出来，整理这些素材的各种初稿包含手稿、幻灯片、大纲，以及形成本书架构草案的工作就落在了 Dena Pauletti 的身上。她是 RBCS 相当出色和严谨的技术主管，没有她及时并且不懈的协助，完成本书恐怕需要延迟数月时间。

当然，没有基础，尤其是 ISTQB 基础级大纲，就不可能有高级测试认证大纲。我很荣幸也曾是基础级大纲工作组的一员，我感谢他们这么多年来出色的工作，正是他们这么多年建立的坚实基础，才有了今天的高级大纲，有了本书的出现。

在本书的教程和素材创作过程中，我运用了我所有在从事“作者”、“测试人员”、“顾问”和“讲师”时积累的经验，因此令我受益的人不计其数，很难一一列出。我感谢那些买我其他书籍的读者们，感谢你们对我“作家”这一角色的鼓励。我感谢那些曾经跟我一起工作的项目组成员，正因为有你们，我才有机会不断提升自己，成为一个合格的测试经理、测试分析师、测试技术分析员。我感谢那些曾经聘请我做顾问，与我共事的人们，是你们给我机会从你们公司学到更多的经验和知识。我还要感谢那些曾经上过我课的

学员们，你们教会我的远比我教给你们的多得多。我感谢我的读者、同事、客户和学生们，希望我所尽的微薄之力能表达我对你们无尽的感谢。

在过去的十多年间，我经营了一家提供软件测试的服务公司——RBCS。现在这家公司已经从一个简陋的小公司成长为一个国际化的集咨询、培训和外包等多种服务于一体的公司，拥有的客户遍布六大洲。这不可能是我个人努力的结果，它持续成功的动力来自于我们一百多名雇员、次级承包商和业务合作伙伴。我感谢你们为客户所做的辛勤工作，没有 RBCS 的成功，我将很有可能没有精力和财力写这些技术相关书籍，因为这些书，能给我带来自豪感，却无法给我带来财富。在此，我再次向你们表示无尽的感激，正是我们共同的努力，才有了今天的成就。

最后，我要感谢我的家人，尤其是我的妻子 Laurel 和我的两个女儿 Emma 和 Charlotte。经营一家国际化的软件测试服务公司意味着我的日程总是排得很满，只有很少的时间可以和我的家人在一起，而我永无止境的写作的习惯使得相聚的时间少之又少。Laurel、Emma 和 Charlotte，我想让你们知道的是，我知道我永远都无法偿还你们曾经给予我的一切，但是我对你们每个人的爱，远多于我们共享的时光。

概述

本书涵盖了软件测试分析师需掌握的关于高级软件测试方面的知识。也就是说，本书论述的内容，是每位选择软件测试为职业的从业人员都需要了解的，但专注介绍软件测试分析、设计、执行和结果评估相关的技术和技能。当然，我假定本书的读者已经知道关于测试工程、测试设计、测试工具、软件开发周期中的测试以及测试管理的基本概念。作为一名专业测试人员，您已经准备好深入理解这些基本概念，并将这些成熟的、有深度的概念应用到您的日常工作中去。

本书遵循国际软件测试认证委员会（ISTQB）的高级大纲，其素材和学习目标专注于高级软件测试分析师部分。就这点而论，本书将对您准备ISTQB高级软件测试分析师模块的认证考试有所帮助。您可以利用本书自学准备考试，或是将其与认证考试相关的在线学习和培训课程结合使用。如果您正在学习 ISTQB 授权的高级软件测试分析师模块的培训课程，本书将是一本理想的辅助教材。

不过，即便您对 ISTQB 考试没有兴趣，仍然可以发现本书会对您可能从事的高级软件测试工作有所帮助。如果您是一位软件测试经理、测试主管、测试分析师、测试技术分析员、自动化测试工程师、手动测试工程师、编程人员或是需要对软件测试有深入理解的其他领域的从业人员，那么本书对您也将很有帮助。

如上述，本书专注于测试分析，由 11 章组成，主要内容如下：

1. 软件测试基础。
2. 测试过程。
3. 测试管理。
4. 测试技术。
5. 软件特性测试。
6. 评审。
7. 事件（缺陷）管理。
8. 标准和测试过程改进。
9. 测试工具和自动化。
10. 个人技能和团队构成。
11. 认证考试准备。

本书的章节是遵从ISTQB的高级大纲的章节编写的，但针对作为软件测试分析师需要掌握的章节内容进行了详细的论述，其他软件测试分析师只需熟悉的章节内容论述则相对简略。例如，本书详细地介绍了软件测试技术，因为这是一名软件测试分析师需要掌握的核心技能，相比之下，其在测试管理上的介绍就少很多。

如果您也读了本书的姊妹篇《高级测试经理》，会发现有一些与本书同名的章节，但是由于侧重点不同，详尽程度也有所不同。例如软件测试分析师需要了解相当多的关于事件管理的知识，因为他们需要花很多时间撰写事件报告，因此需要了解如何将其做好。测试经理也需要知道很多关于事件管理的知识，但是他们更多的需要把重点放在如何使事件在其从被报告到被解决的生命周期中得以持续推进，以及如何从事件报告中提炼出有效的度量元。

一名软件测试分析师应该具备什么技能？或者，换句话说，您可以通过本书学习或者提高什么技能？

- 能够制定并应用基于业务领域需求的测试策略。
- 能够基于用户的质量期望进行系统分析并应用于测试。
- 能够评估系统需求是否满足业务目标。
- 能够充分准备和执行测试活动，并报告这些活动的进度。
- 能够提供必要的证据和数据支持评估及其结果。
- 能够使用必要的工具和技术以实现预先设定的目标。

本书将会侧重于这些主要的概念，建议您记住这些概要性目标，因为这些目标将会贯穿于本书的各个章节。

在撰写本书以及关于高级测试管理的姊妹篇时，我一直牢记一个很重要的问题，如何使得学习资料对您有用。如果您正在使用本书准备ISTQB高级软件测试分析师的认证考试，那么建议您先阅读本书的第11章，然后再依次阅读其他10章，如果您并没有打算参加该考试，而只是希望通过学习本书拓展您关于测试的知识面，那么建议您只阅读本书的1~10章即可。当然如果您只是想将此书作为一本参考书，那么就阅读那些您感兴趣的章节。

前10章中的每一章将会被分为若干个小节。大部分的划分方式沿用了ISTQB高级大纲中的结构组织，但是有一些子章节的划分与大纲不尽相同。您会发现每个章节的开始都会有一个深色背景的文字框，以描述本章节的学习目标。如果您有兴趣知道每个学习目标前面K2、K3、K4这些标注，以及ISTQB大纲中学习目标的组织与含义，请阅读第11章。

在很多方面，软件测试与弹钢琴、烹饪、开车很相似。为什么？因为无论以上何种活动，您都需要先通过阅读书籍学习相关知识，但直到实际操作之前，这些知识都所用甚微。所以我在关键的概念中加入了很多实际、实用的练习。我鼓励您通过本书中的练习实践学到的概念，并将这些理念融会贯通到自己主持或参与的测试项目中去。只有真正的“做”软件测试，您才可能成为一名专业的高级软件测试人员。

关于作者

Rex Black 先生在软件工程和系统工程领域有 25 年的从业经验，他是 RBCS（www.rbcs-us.com）的总裁，也是软件、硬件和系统测试的领军人物。经过十多年的发展，RBCS 的服务覆盖了咨询、外包、软件/硬件测试培训等多个领域。RBCS 拥有一批业界最有经验和知名度的顾问，帮助其客户开展产品测试，组建测试团队，提高其测试水平，同时为全球范围内的数百个客户提供测试人力资源。RBCS 服务的对象不仅有《财富》20 强企业，也有一些刚起步的公司，它帮助客户通过改善产品开发的过程，节省时间和金钱，减少在技术支持上的投入，提高公司的信誉度等。

作为 RBCS 的领导者，Rex 是当今在软件测试领域最多产的作者，他最受欢迎的一本书 Managing the Testing Process 在全球售出超过 30 000 本，并发行了日语、汉语和印度语的译本。他的其他 3 本关于测试的书 Critical Testing Processes、Foundations of Software Testing 和 Pragmatic Software Testing，也售出了数万本，包括希伯来语、印度语、汉语、日语和俄语版本。他撰写了超过 25 篇测试领域的文章，出席了数百个专业论文的讲习班和研讨会，并三十多次在全球各大会议和活动中担任专题演讲嘉宾。与此同时，Rex 还兼任了国际软件测试认证委员会（International Software Testing Qualifications Board，ISTQB）的主席和美国软件测试认证协会（American Software Testing Qualifications Board，ASTQB）的理事。

目录

第 1 章　软件测试基础 …… 1

1.1　概述 …… 1
1.2　软件生命周期中的测试 …… 2
1.3　特定系统 …… 5
1.4　度量元和度量 …… 8
1.5　职业道德 …… 10
1.6　认证考试模拟题 …… 11

第 2 章　测试过程 …… 13

2.1　概述 …… 13
2.2　测试过程模型 …… 14
2.3　测试计划和控制 …… 14
2.4　测试分析和设计 …… 14
2.4.1　功能测试目的 …… 16
2.4.2　测试准则 …… 18
2.4.3　标准 …… 20
2.4.4　静态测试 …… 21
2.4.5　度量元 …… 22
2.5　测试实现和执行 …… 23
2.5.1　测试规程的准备工作 …… 23
2.5.2　测试环境的准备工作 …… 25
2.5.3　混合型测试策略 …… 26
2.5.4　测试执行的开始 …… 26
2.5.5　一个独立的测试规程的运行 …… 27
2.5.6　测试结果的记录 …… 28
2.5.7　利用业余测试员 …… 29
2.5.8　标准 …… 29
2.5.9　度量元 …… 32

2.6 评估出口准则和报告……32
2.6.1 测试套件总结……33
2.6.2 缺陷的分解……34
2.6.3 确认测试未通过率……35
2.6.4 系统测试出口评审……36
2.6.5 标准……37
2.7 评估出口准则和报告练习……37
2.7.1 系统测试退出评审……38
2.8 评估出口准则和报告练习解答……39
2.9 测试结束活动……42
2.10 认证考试模拟题……42

第3章 测试管理……44

3.1 概述……44
3.2 测试管理文档……45
3.3 测试计划文档模板……46
3.4 测试估算……46
3.5 测试进度计划……46
3.6 测试进度监视与控制……47
3.7 测试的商业价值……47
3.8 分布式测试、外包测试和内包测试……47
3.9 基于风险的测试……48
3.9.1 风险管理……50
3.9.2 风险识别……51
3.9.3 风险分析或风险评估……52
3.9.4 风险减轻或风险控制……53
3.9.5 关于风险识别和评估结果的举例……56
3.9.6 软件生命周期中的基于风险的测试……57
3.9.7 （具备）风险意识的测试标准……57
3.10 基于风险的测试练习1……58
3.11 基于风险的测试的练习解答1……60
3.11.1 项目风险的副产品……62
3.11.2 需求缺陷的副产品……63
3.12 基于风险的测试练习2……63
3.13 基于风险的测试练习解答2……63
3.13.1 测试用例排序原则……64
3.14 失效模式和影响分析……65
3.14.1 测试管理的相关问题……65

3.15 认证考试模拟题 ····65

第 4 章 测试技术 ····67

4.1 概述 ····67
4.2 基于规格说明的测试技术 ····69
4.2.1 等价类划分 ····71
4.2.2 避免错误的等价类划分 ····73
4.2.3 用等价类划分技术生成测试用例 ····73
4.2.4 等价类划分技术练习 ····76
4.2.5 等价类划分技术练习题答案 ····76
4.2.6 边界值分析 ····78
4.2.7 等价类划分技术和边界值分析的样例 ····79
4.2.8 有多少边界值 ····87
4.2.9 边界值练习 ····88
4.2.10 边界值练习解答 ····89
4.2.11 决策表 ····93
4.2.12 在表格中收缩列项 ····95
4.2.13 因果图 ····97
4.2.14 将决策表测试与其他技术结合 ····98
4.2.15 决策表中的非排他性规则 ····100
4.2.16 决策表练习 ····100
4.2.17 决策表练习解答 ····101
4.2.18 用例 ····106
4.2.19 用例练习 ····111
4.2.20 用例练习解答 ····111
4.2.21 基于状态的测试与状态转换图 ····116
4.2.22 状态和子状态 ····120
4.2.23 状态转换表 ····121
4.2.24 切换覆盖 ····124
4.2.25 用其他技术进行状态测试 ····126
4.2.26 状态测试练习 ····127
4.2.27 状态测试练习解答 ····127
4.2.28 结对测试 ····137
4.2.29 结对测试练习 ····141
4.2.30 结对测试练习解答 ····142
4.2.31 分类树 ····144
4.2.32 分类树练习 ····146
4.2.33 分类树练习解答 ····147

4.2.34 根据测试依据设计测试……148
4.2.35 根据测试依据设计测试的练习……151
4.2.36 根据测试依据设计测试的练习解答……151
4.3 基于结构的测试技术……153
4.3.1 基于缺陷和基于经验的技术……154
4.3.2 缺陷分类……155
4.3.3 错误推测……157
4.3.4 检验表测试……158
4.3.5 探索性测试……160
4.3.6 测试章程……161
4.3.7 软件攻击……162
4.3.8 一个有效攻击示例……165
4.3.9 其他攻击方式……166
4.3.10 共同的主题……167
4.4 基于缺陷的和基于经验的测试技术练习1……168
4.4.1 基于缺陷的和基于经验的测试技术练习解答1……169
4.4.2 基于缺陷的和基于经验的测试技术练习2……169
4.4.3 基于缺陷的和基于经验的测试技术练习解答2……170
4.5 静态分析……172
4.6 动态分析……172
4.7 认证考试模拟试题……172
第5章 软件特性测试……179
5.1 概述……179
5.2 领域测试的质量属性……180
5.2.1 功能的准确性……181
5.2.2 功能的适应性……181
5.2.3 功能的互操作性……184
5.2.4 功能的互操作性练习……185
5.2.5 功能互操作性练习解答……186
5.2.6 功能安全性……188
5.2.7 可达性……189
5.2.8 可用性……190
5.2.9 可用性练习……193
5.2.10 可用性练习解答……193
5.3 技术测试的质量属性……194
5.3.1 技术安全性……195
5.3.2 安全性攻击……196

5.3.3 可靠性 ……199
5.3.4 效率测试 ……202
5.3.5 可维护性测试 ……204
5.3.6 可移植性测试 ……204
5.4 认证考试模拟试题 ……206

第 6 章 评审 ……208

6.1 概述 ……208
6.2 评审的原则 ……210
6.3 评审的类型 ……212
6.4 评审的引入 ……215
6.5 评审的成功因素 ……216
6.5.1 Wiegers 的评审检查表 ……218
6.5.2 Deutsch 的评审检查表 ……220
6.6 Wiegers 检查表评审练习 ……221
6.7 Wiegers 检查表评审练习解答 ……221
6.8 Deutsch 检查表评审练习 ……223
6.9 Deutsch 检查表评审练习解答 ……224
6.10 认证考试模拟题 ……225

第 7 章 事件管理 ……226

7.1 概述 ……226
7.2 何时可以发现一个缺陷 ……227
7.3 缺陷生命周期 ……227
7.4 缺陷要素 ……231
7.5 度量元和事件管理 ……234
7.6 事件沟通 ……235
7.7 事件管理练习 ……235
7.8 事件管理练习解答 ……236
7.9 认证考试模拟题 ……237

第 8 章 标准和测试过程改进 ……239

第 9 章 测试工具和自动化 ……241

9.1 概述 ……241
9.2 测试工具的概念 ……242
9.2.1 测试自动化成本 ……243
9.2.2 测试自动化风险 ……243

9.2.3 测试自动化收益……244
9.2.4 测试自动化策略……245
9.2.5 测试工具集成和脚本……245
9.2.6 测试工具分类……247
9.3 测试工具类型……248
9.3.1 测试管理工具……248
9.3.2 测试执行工具……249
9.3.3 调试、故障排除、缺陷散播和缺陷注入工具……250
9.3.4 静态和动态分析工具……251
9.3.5 性能测试工具……252
9.3.6 Web 测试工具……254
9.3.7 模拟器和仿真器……254
9.4 认证考试模拟试题……255

第 10 章 个人技能和团队组成……256

10.1 概述……256
10.2 个人技能……257
10.3 测试团队整体实力……257
10.4 组织的适合测试……257
10.5 激励……257
10.6 沟通……258
10.7 认证考试模拟试题……259

第 11 章 认证考试准备……260

11.1 学习目标……260
11.1.1 级别 1：牢记（K1）……260
11.1.2 级别 2：理解（K2）……261
11.1.3 级别 3：应用（K3）……261
11.1.4 级别 4：分析（K4）……261
11.1.5 学习目标级别的由来……262
11.2 ISTQB 高级认证考试……262
11.2.1 场景分析考题……263
11.2.2 考题的演变……264

附录 A 参考文献……266

附录 B HELLOCARMS 下一代房屋净值贷款系统……269

B.1 系统需求文档……269

B.1.1 版本……269
B.1.2 术语表……269
B.2 概述……270
B.2.1 非正式用例……271
B.2.2 范围……271
B.2.3 系统商业利益……272
B.2.4 系统功能需求……272
B.2.5 系统可靠性需求……275
B.2.6 系统易用性需求……275
B.2.7 系统效率需求……276
B.2.8 系统可维护性需求……276
B.2.9 系统可移植性需求……277
B.3 致谢……277

附录 C 问题答案……278

pin51.com 拼吾爱

第1章 软件测试基础

> “我是谁？我为什么会在这里？”
>
> 引自美国副总统候选人、海军上将 James Stockdale
> 于 1992 年副总统竞选中的演讲

高级大纲的第 1 章将会把重点放在一些与以后章节紧密相关的内容及背景资料上，本章包括以下 5 个小节：

1. 概述。
2. 软件生命周期中的测试。
3. 特定系统。
4. 度量元和度量。
5. 职业道德。

下面通过对每个章节的深入论述来看看它们与软件分析的关系。

1.1 概　　述

学习目标：

能回想起该部分内容即可。

本章将介绍一些与测试从业人员普遍相关的基础知识，主要包括以下 4 个方面：

- 软件生命周期及其对测试的影响；
- 特定系统及其对测试的影响；
- 对于测试与质量的度量元和度量；
- 职业道德问题。

可能因为这一章标题的缘故，您会误认为这些内容与那些已经在基础级大纲中出现过的内容重复，但事实上以上所提到的概念和观点是基础级大纲相关内容的延伸，并将会在后面的章节中更详细地论述。

pin51.com 最新最全 拼吾爱 下载分享

ISTQB 术语

软件生命周期：从开始构思软件产品到产品不再被使用为止的时间周期。软件生命周期通常包括概念构思阶段、需求分析阶段、设计阶段、实现阶段、测试阶段、安装和验收阶段、运行和维护阶段，有时还包括退役阶段。注意：这些阶段有可能重叠或被迭代。

1.2 软件生命周期中的测试

学习目标：

能回想起该部分内容即可。

基础级大纲中的第 2 章论述了如何将测试与软件生命周期相结合。与基础级大纲一样，在高级大纲中，您需要了解，成功的测试必须要融合到软件生命周期中，无论软件生命周期的类型是顺序型、增量型、迭代型，还是螺旋型。

在软件生命周期中，测试过程和其他过程恰当地融合是成功的关键。这种融合在测试与其他过程活动的关键接口及相互切换中显得尤为必要，例如以下这些过程活动：

- 需求工程和管理；
- 项目管理；
- 配置与变更管理；
- 软件开发与维护；
- 技术支持；
- 技术文件。

让我们来看两个关于“联盟（Alignment）”的例子。

在顺序型软件生命周期模型中，一个关键的假设是，项目组会在项目初期定义好需求，并在之后的项目周期中管理这些需求的变更（但愿变更的数目有限）。在这一前提下，如果项目组遵循一个规范的需求流程，系统级别的测试由一个独立的测试团队负责，那么，（团队）可以采用基于需求分析的测试策略。

在顺序型软件生命周期模型中应用这一测试策略，测试团队可以在项目的初期就参与进来，通过分析需求规格开始测试计划和设计来识别测试条件。在测试计划、分析和设计的过程中，测试团队可以直接发现需求中的缺陷，使测试成为一种缺陷预防活动。对失效的检测则在软件生命周期的后续阶段，系统测试执行阶段时开始。

然而，如果项目依据某一敏捷开发方法，例如 Scrum，采用增量型软件生命周期模型，测试团队往往很难在项目的早期收到全部的需求，相反的，测试团队会在每 30 天的 sprint 的初期收到需求。

测试团队在此情况下所能采取的最理想的行动就是先识别关键的质量风险并对其进行优先级划分，而不是急于对尚不完整的需求进行分析。也就是说，团队可以采用基于

风险分析的测试策略。基于此测试策略，仅在测试执行前才进行针对性的测试设计和开发，速度虽快却潜在地降低了测试的缺陷预防效应。对失效的检测则在项目早期，在第一个 sprint 末期就开始了，并持续贯穿在其后的每一个短的子周期末期。在这种情况下，主测试过程的测试活动会彼此之间并与软件生命周期中的其他主要活动发生重叠或者并行进行。

对测试来讲，无论采用何种软件生命周期模型，尤其是节奏较快的敏捷型软件生命周期模型，有效的需求和配置管理是至关重要的，缺乏恰当的需求管理将导致测试团队无法了解系统是什么，系统能做什么；而缺乏好的配置管理，正如在基础级大纲中论述过的，将会导致一些需求变更的缺失，以致相关人员将无法及时了解什么时候测试过什么，也就很难透彻地理解测试的结果。

ISTQB 术语

综合系统（system of systems）：多异构，被嵌入在多层次多领域解决相互关联的大型跨学科共同问题和目的的网络上的分布式系统。

基础级大纲中引用了 4 个典型的测试级别：

- 单元或组件；
- 集成；
- 系统；
- 验收。

基础级大纲还提到了这些测试级别中的一些变体，尤其是在集成和验收测试级别。

集成测试可以理解为组件集成测试，将一系列组件集成为一个系统，在整个测试过程中对各个构建（build）进行测试，当然它也可以被理解为系统集成测试，将一系列系统集成为一个综合系统，测试对象即是这些系统的集合体。

在基础级大纲中我们已经论述过，验收测试的变体包括用户验收测试和常规验收测试。

关于这 4 个测试级别以及它们的变体，在高级大纲中，您需要知道另外一些可能会在项目中出现的级别：

- 软/硬件集成测试；
- 特性交互测试；
- 用户产品集成测试。

并且（在高级大纲中）对于每个测试级别都应该包含下列全部或者大部分内容：

- 定义清晰的测试目标和范围；
- 对测试依据具备可追溯性（如果能够的话）；
- 各个测试级别和软件生命周期各个阶段的入口准则及出口准则；
- 测试交付物，包括测试结果报告；
- 针对不同的测试级别、测试团队和系统内在的风险，采用的测试技术；

pin51.com 最新最全源码资源下载 拼吾爱

- 度量元和度量；
- 按实际需求适用于各个测试级别的测试工具；
- 按实际需求遵从组织或其他标准。

当 RBCS 的合作伙伴对其测试团队进行评估时，我们通常会发现他们会按照各个测试级别进行评估，但这些测试都是彼此独立进行的，这往往会导致效率不高，且令人困惑，这些话题将在本书的姊妹篇《高级测试管理》中做更多阐述。软件测试分析师应该牢记要善于使用测试文档，例如测试方针，在各个测试级别保持与测试相关人员的密切交流与配合，以减少测试结果的缺失、重叠和模糊。

以 V 模型（如图 1-1 所示）为例，将系统测试级别设为论述的对象，进一步来分析“联盟”的概念。

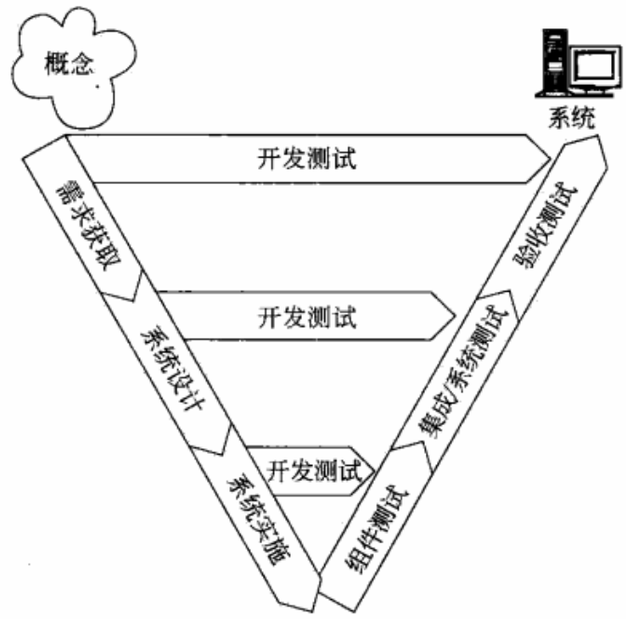


图 1-1 V 模型

在 V 模型中，一个恰到好处的联盟测试过程应该是测试计划与项目计划同步进行，换言之，项目一开始，就应该立即引入测试。

一旦测试计划被审批通过，测试控制过程便开始了，直至测试结束。测试分析、设计、实现、执行、评估出口准则、生成测试结果报告，都将按照测试计划的约定进行，任何与计划的偏差也配有相应的管理措施。

测试计划结束后，测试分析就要立即开始了，有时两者甚至是同步进行的。而测试分析和测试设计与系统需求（分析）、概要设计、详细设计是同步进行的。测试实现包括测试环境的搭建，将起始于系统设计阶段，持续至测试执行开始之前。

当满足测试入口准则时，即开始测试的执行部分。现实中，大多数的情况是，当大部分入口准则满足，不能满足的个别准则可以被放弃的情况下，也可以开始执行测试。在 V 模型中，系统测试执行的入口准则通常是，组件测试和集成测试均已完成，并且符合相应的出口准则。再次强调，上述一些出口准则也会视实际情况而被放弃。

而对出口准则的评估和测试结果的记录，将会贯穿于测试执行过程的始终。

在测试执行宣告完成后，即启动相应的测试结束活动。

如上述这样将各项测试活动和系统软件生命周期其他活动精确联盟绝非一蹴而就。没有先期（对联盟策略）的思考和策划，测试活动是无法渗透、贯穿到整个软件生命周期的各个过程的。

另一方面，对于各个测试级别，不管选择何种软件生命周期模型和测试过程，测试经理都需要贯彻联盟策略。测试经理不仅是要在测试和项目计划期间，更要在测试控制过程中进行联盟。

无论选择何种测试过程和软件生命周期模型，每个项目都有其自身的“怪癖”（特定问题）。尤其是一些复杂的系统，例如普遍存在于军方客户和 RBCS 的大客户的综合系统。这种情况下，测试经理不仅要计划联盟测试过程，更要灵活运用联盟策略以适应实际情况。无论是对测试还是整个软件生命周期来讲，一些现成的过程模型，都无法适用于这种复杂的项目。

1.3　特 定 系 统

学习目标：

能回想起该部分内容即可。

综合系统是为完成一个共同目标的相对独立的几个系统绑定在一起形成的。因为这些系统相对独立，但又绑定在一起，使得该类型的系统往往缺乏一个独立、清晰的用户或操作接口，没有统一的数据模型、可兼容的外部接口等。

这类项目包括以下特性和风险：

- 商用现货软件（COTS）与一些定制开发的软件的整合往往需要花费很长时间。
- 显著的技术、软件生命周期和组织的复杂性及差异性，组织和软件生命周期的复杂性包括保密性、公司机密和规章制度等问题。
- 软件内包、外包、离岸开发时，常常出现的不同软件生命周期中的开发过程和其他过程分属不同团队（复杂交错带来的问题）。
- 由内部系统耦合引发的严重的潜在的可靠性问题，一个本身就存在弱点的系统可能会在整个综合系统中产生一系列的连锁失效反应。
- 系统集成测试包括互操作性测试变得至关重要，因此定义清晰的测试接口是必要的。

很显然，综合系统远比单个系统复杂得多，这就对组织、技术、过程以及团队的智慧都有了更高的要求。当一个项目的规模和复杂度越大，良好的项目管理，规范的软件开发生命周期及其过程管理、配置管理和质量保证体系就更重要。

在这里让我们着重看一下软件生命周期的含义。

如上所述，在综合系统项目中，我们通常需要在多个级别进行集成。需要先将多个组件集成为系统，再将系统集成为我们需要构建的综合系统。

pin51.com 最新最全 拼吾爱

我们通常还需要多版本管理和控制系统和过程，除非所有的系统是由一个公司（可想而知会是很大的公司）开发的，而在该公司的软件开发团队中都遵守相同的流程。顺便提一句，我和我的合作伙伴在对一些大型公司进行评估时，这种情形也并不常见。

当然，项目的周期也将会变得很长，我曾经见过一个项目用了5～7年的时间来计划一个项目。如果一个包含5或6个系统的综合系统项目，“仅”花费40或50个人1年的时间，可以认为其时间跨度已经相对短，规模也相对小。即使在这样的项目中，也往往需要有多个不同的团队负责多个级别的测试。

因为项目规模和复杂度的原因，很容易会因为切换或移交责任处理不当而造成系统崩溃。所以我们需要在项目成员之间建立规范的信息移交机制，尤其是对于项目阶段性总结期间的里程碑事件的妥善移交。

甚至在仅仅对现货系统集成时，这些系统本身也是在不断更新演进中的。对于那些定制系统经常会面临这样的情况。所以，管理层将要面临如何协调各个系统的开发的挑战，测试分析师则会面临在综合系统的系统集成测试级别发生变更时，如何进行合理的回归测试的挑战。

尤其是现货系统，有时会在没有任何警示的情况下，由一些外部实体和事件触发维护测试，例如报废、破产或某个系统升级。

在一个综合系统项目中，其基本的测试过程不是二维的，而是一个金字塔结构，如图1-2所示。

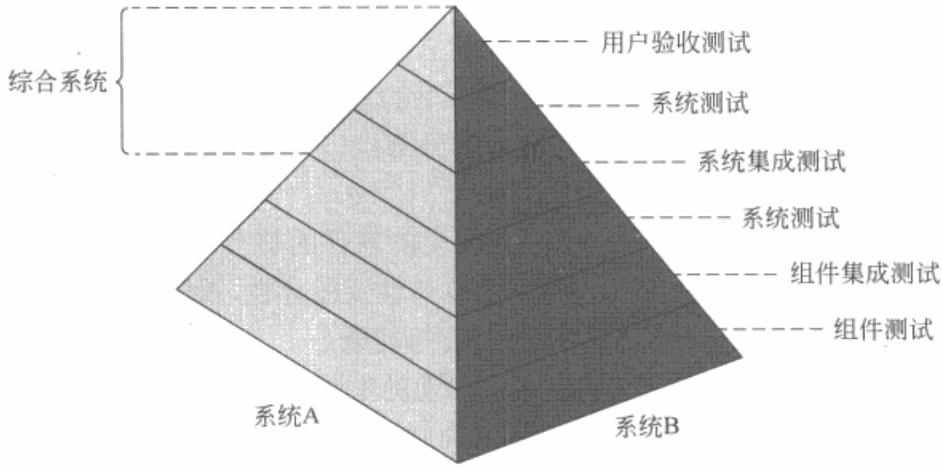


图1-2 综合系统项目的基本测试过程

金字塔的最下层是组件测试，每个系统都要独立地进行组件测试。

沿着金字塔向上，就是组件集成测试，每个系统都要独立地进行组件集成测试。

接下来是系统测试，同样的，每个系统都要独立地进行系统测试。

请注意，对于上述3个测试级别，很可能每个系统属于不同的供应商，即使来自同一个供应商，也很可能是由不同的团队协作完成的。

再向金字塔的上面看，到系统集成测试。现在，我们终于提到一个能贯穿所有系统

的测试级别。再向上，是系统测试，侧重在贯穿整个系统的终端到终端测试。最后进行用户验收测试。在这些测试级别的测试，多是由同一个公司负责，虽然也可能由同一个公司的不同的团队协作完成。

> **ISTQB 术语**
>
> **安全关键系统**：一个系统的失效或故障可能导致人员死亡或受到严重伤害，设备丢失数据或严重损坏，或环境损害。

简单来讲，安全关键系统往往是与使用者或者受众生命攸关的系统。那么，一旦某个系统有所缺陷，甚至是性能或可靠性暂时性的降低，或是由于采取某些行动而产生的不良副作用，都有可能对人造成伤害甚至夺取人的生命，或者，就军事系统而言，在战斗的关键时刻无法杀伤敌人。

安全关键系统和综合系统一样，存在一定的特点和风险：

- 因为缺陷会造成死亡，死亡会引发民事或刑事责任。事实证明，通常充分的测试可以降低这一可能性。
- 显而易见地，各种各样的规范和标准常被应用于安全关键系统中，这些规范和标准能够约束过程、组织架构和产品。与以往其他约束条件不同，这些规范和标准将专注于提高质量水平，而不是通过牺牲质量而获得进度、预算、功能交付物之间的平衡。总之，质量保证（在此类项目中）具有极其重要的优先级。
- 通常，对于开发和测试都有一套严格的方法。在软件生命周期中，从监管要求到测试结果都要保持其可追溯性，这就需要提供一些相关的合规性证明（Demonstrating Compliance），需要有全面的、具体的文档来支撑，而不需要概要性的审计，即使这些全面、具体的文档来自于一个非测试领域的（审计）专家。

要想推行规范，审计是很普遍的手段。合规性证明可包括从监管要求追查到开发再到测试结果，通常是由第三方机构进行审计。无论从人员或是过程的角度来看，前期的可追溯性证据的收集和后期的审计、合规性证明活动都会对管理、开发、测试和主管部门产生一定的影响。

在软件生命周期中，通常是早在设计阶段，项目团队就应该运用安全分析技术识别潜在的风险，单点失效通常可通过系统冗余被逐个排查解决。

在某些情况下，安全关键系统是很复杂的系统，甚至是综合系统。而有些情况下，一些非安全关键的组件或系统会被集成到安全关键系统或综合系统中。例如：互联网或通信设备其本身并不是安全关键系统，但是如果要将其集成到应急系统或军用系统中，就变成了安全关键系统的一部分。

这些情况下，规范的质量风险管理变得尤为重要。幸运的是，目前已经有很多这样成熟的技术，如失效模式和影响分析（FMEA）、失效模式影响及危害性分析（FMECA）、危险分析和软件共因失效分析。

pin51.com 最新最全源码资源下载 拼吾爱

1.4 度量元和度量

学习目标：

能回想起该部分内容即可。

本书我们利用度量元和度量建立一些期望值，并利用这些期望值，指导我们的测试。您可以且应该在整个软件生命周期中应用这些度量元和度量。因为使用良好的度量元和度量，并与项目的目的和目标相结合，测试分析师能够以一种连贯的和一致的方式跟踪和报告测试和质量结果。

ISTQB 术语

度量元：度量的计量单位以及采集方法。

度量计量单位：约束数据分析类型的标准。

度量：给实体赋予一个数值或类别以描述其某个属性的过程。

测量：度量时赋予实体某个属性的数值或类别。

缺乏度量元和度量不仅会直接导致对质量和测试的主观臆断，还将会引发在整个测试过程中对测试结果含义的不断争议，甚至会由于这些测试结果的取值含混不清而影响团队之间的交流，降低测试的效力和效率。

我们不但需要度量元和度量，还需要一个基线。什么是所谓“好”的测试结果？什么是可接受的测试结果？什么是不可接受的测试结果？没有清晰定义的基线，就无所谓测试的成功。事实上，在 RBCS 对客户进行评估时，比对测试团队效力和效率的度量元定义不明确更常见的问题，就是没有基线导致的对测试结果错误的或者是不切实际的期望值（这些期望值根本无法实现）。

什么可被定义为一个度量元，并可应用于度量（过程）来追踪（收集数据）的话题是无止境的，以下的例子可供参考：

- 计划进度表及计划覆盖率；
- 需求及其与测试相关的进度、资源和任务；
- 工作量和资源的使用情况；
- 里程碑和测试范围；
- 计划成本和实际成本；
- 风险，包括质量风险和项目的风险；
- 缺陷，包括发现缺陷总数、已修正缺陷总数、当前积压缺陷数、平均（缺陷）关闭周期以及缺陷按配置、子系统、优先级或严重性的分布。

在测试计划期间，我们首先要确定期望值，也就是之前所提到的“基线”。在测试控制中，我们可以对比这些期望值，测量实际结果及其趋势。在记录测试结果时，我们可以根据客观的、事先已达成一致的度量元（采集到的数据）以及其可协助达到的切合实

pin51.com

际的目标，连贯地向管理层说明关于过程、产品和项目的各个重要方面的信息。

在制定一个测试度量元和度量体系时，需要考虑 3 个主要方面：定义、追踪和报告。让我们首先从定义开始。

在一个成功的测试度量元体系中，首先需要为项目定义一系列有效、准确和简明的质量和测试的度量元。要避免使用太多的度量元。事实证明，度量元的数量太多将会加大度量元的使用难度，甚至需要付出过高的测量成本。这使本来应该对观测者或利益相关者有所启迪的度量元，反而会令人迷惑不解。

其次，还需要确保这些度量元含义的一致性（无二义性），以减少在度量元的结果、分析和趋势上无谓的争执和分歧。一个人人都在其定义上有分歧的度量元体系是没有意义的。

最后，按照一个流程或任务、组件或系统、个人或团队的阶段性目标和最终目标定义度量元。Victor Basili 提出的著名的“目标—问题—度量元（Goal Question Metric）”技术，为如何制定有意义的度量元提供了方法，借助这一技术，我们由工作的最终目标开始（在这里，工作是指测试），到因需要了解“如何满足这个目标”而提出的问题，再到最终的——特定的度量元的定义。

举例来讲，测试的一个典型目标就是建立信心。那么自然而然会产生一个问题，系统已经被测试了多少？而对于覆盖率的度量元包括：测试覆盖需求的百分比、测试覆盖的分支和语句的百分比、测试覆盖的接口的百分比和测试覆盖的风险的百分比等。

让我们再来看追踪。

因为追踪是在度量体系中重复出现的活动，使用自动化的工具可以减少捕获缺陷、跟踪、分析、记录和测量度量元的时间。

请务必对特定的度量元随着时间的推移加以客观和主观的分析（更新），尤其是在其出现多义性的趋势时，尽量避免急于下结论，或者是直接将度量元交付给其他人，并鼓励他们也这样做。

要洞察并控制个人的主张或倾向对度量元及其度量含义的影响。每个人都认为自己很客观，但是个人的喜好往往会影响到他们做出的结论。

最后，来看看报告。

最为重要的是，关于度量元和度量的报告应该令管理层和利益相关者有所启发，而不是迷惑或是误导他们。在一定程度上，只有对度量元巧妙地定义和仔细地追踪，这一目标才有可能达成；但是也有可能度量元相当地清晰和有意义，但是稍为逊色的表达仍有可能迷惑别人。Edward Tufte 针对这个内容编写了系列著作，其中 The Graphical Display of Quantitative Information 提供了关于如何绘制高质量图表的方法。[1]

好的基于度量元的测试报告应该是易懂、简洁明了的。测试报告需要依靠重要的信息而非琐事吸引您的注意力。只有这样，一份好的基于度量元和度量的测试报告才会帮助管理者引导项目走向成功。

1　我读过 3 本与这一话题有关的 Tufte 的书，并在此强烈推荐，The Graphical Display of Quantitative Information、Visual Explanations 和 Envisioning Information。

pin51.com 最新最全 拼吾爱

并不是每一种度量元的图形显示效果都是一样的，或者一样有效。某一时刻一组及时传递的数据的快照，有可能是表达某些信息的最佳方式，例如，对于那些关键质量风险的计划测试与实际测试的覆盖率。而随时间变化的趋势图，可能比较适合表达其他信息，例如，从测试开始，发现缺陷总数和已修正缺陷总数。对于因果关系分析图，可能比较适合表达一些静态的信息，例如，点阵式图表可用以表达测试工程师的经验和缺陷报告被驳回的相关性（或非相关性）。

1.5 职业道德

学习目标：

能回想起该部分内容即可。

ISTQB 术语

职业道德：ISTQB 未提供该术语的定义。

很多职业都有其职业道德标准。从专业的字义上来讲，职业道德是指“一组被认可的关于特定类别的人类活动，或是一个特定的团队、文化等的行为规范”。[2]

由于作为一个测试分析师，通常会有机会接触到一些机密和需特殊授权的重要信息，职业道德可以帮助您恰当地运用这些信息。在此补充说明一句，您需要按照职业道德选择既定情形和局限性下，可能达到的最好的行为和结果。注意“可能达到的最好的”是指对所有人，而非仅指测试分析师本人。

在这里让我来给出一个关于职业道德的具体行为的例子。我是 RBCS、RBCS NZ 和 PureTesting 3 个相互关联的国际软件测试顾问公司的总裁；同时，我还是国际软件测试认证委员会（ISTQB）和美国软件测试认证委员会（ASTQB）的董事。这样来讲，我应该有很多我们在软件测试顾问方面的竞争对手没有的关于 ISTQB 体系的内部资料。

有时候，我不得不向人们澄清我的商业兴趣促使我帮助编写这些测试大纲，而且我是获得许可来参与编写的。我协助编写了（ISTQB）基础级和高级大纲。

有时候，例如帮忙编写这些大纲，我需要向人们澄清我的商业利益（与此会有潜在的冲突），并得到许可之后，我才帮助编写了基础级大纲和高级大纲。

另外一个例子是关于认证考试考题的，我同意与我在美国软件测试认证委员会的同事都不参加考题的编写。直接介入考题编写会有意无意地使我们的培训资料陷入教考不分（既当运动员又当教练员）的尴尬境地。

作为测试工程师，在您的职业生涯中，会有越来越多的机会显示出您恪守或者缺乏职业道德的本性，培养强烈的职业道德是个永远都不晚的话题。

在 ISTQB 的高级大纲中明确指出，ISTQB 希望所有证书持有者遵守以下职业道德

2 来自 dictionary.com。

法规：

公共（PUBLIC）——认证软件测试人员的行为应与公共利益保持一致。例如，如果你正在测试一个安全关键系统，如果被要求悄悄删掉一些测试缺陷报告，这就是一个涉及职业道德的问题。

客户和雇主（CLIENT AND EMPLOYER）——认证软件测试人员在保证公共利益的前提下，最大限度地保证客户和雇主的利益。例如，如果你知道雇主某个比较重要的项目正处于困境，你卖空持有的股票，然后将关于该项目的信息泄露在互联网上，这绝对是一种职业道德丧失的行为，还很有可能是一种犯罪行为。

产品（PRODUCT）——认证软件测试工程师应保证他们发布的在测试产品和系统中的版本最大限度地符合专业标准。例如，如果你作为顾问，略去了测试计划中重要的细节，使得客户不得不在下个项目继续雇用你，这也是一种丧失职业道德的行为。

判断（JUDGMENT）——认证软件测试工程师应在其提供的专业的判断中保持公正性和独立性。例如，如果项目经理因为某些潜在的商业赞助商的反应，就要求你不要记录在某些方面的缺陷，这是对你独立性的打击，如果你妥协的话也是一种丧失职业道德的行为。

管理（MANAGEMENT）——认证软件测试管理人员和测试领导者应同意提供合乎道德要求的测试管理。例如，只因为你想与某个测试工程师的姐妹建立一段浪漫的关系，而偏爱这个测试工程师，这是严重的职业道德丧失。

专业（PROFESSION）——认证软件测试工程师应致力于提高职业的公正性和信誉并与公共利益保持一致。例如，如果你有机会向你的孩子的同学或是你配偶的同事解释你是做什么的，那么请以你的职业为荣，并向他们说明软件测试是怎样让社会受益的。

同事（COLLEAGUES）——认证软件测试工程师应在工作中热切地支持他们的同事并促进与软件开发人员的合作。例如，靠操纵测试结果使得你不喜欢的软件开发工程师被解雇，也是不遵守职业道德的表现。

自身（SELF）——认证软件测试工程师应终生学习并不断促进职业实践的提升。例如，参加培训课程，读书和在会议中演讲来分享你是如何提升自身和专业技能的。在提升自身的同时又培养自身的职业道德，这是非常符合职业道德的。

1.6　认证考试模拟题

在每个章节的最后，可以试着做几道认证考试模拟题以加强对材料的记忆和理解，准备 ISTQB 高阶测试分析师的考试。

1. 假设你是银行里的一名测试分析师。在用户验收测试阶段中，软件测试分析师通常和用户接触的比较多。银行从一个大的软件供应商那里采购了两套商用现货（COTS）金融软件，过去的资料显示，这两个金融软件的供应商提供高质量的软件，但分别独立开发自己的软件，而该银行是第一次试图把这两套软件集成入自己的系统的客户。那么你认为应该参加以下哪（几）个级别的测试？（注：正确答案不止一个。）

 A　组件测试

pin51.com 最新最全源码分享下载 拼吾爱

B 组件集成测试
C 系统集成测试
D 验收测试

2. 以下哪种观点对于安全关键系统是正确的？
A 它们是由多个 COTS 软件组成的
B 它们是复杂的综合系统
C 它们是可能对人造成伤害甚至危及人的生命的系统
D 它们是军用或智能的系统

pin51.com 拼吾爱

第2章

测试过程

> 把青柠汁放进椰子汁，然后把它们一起喝下去，
> 把青柠汁放进椰子汁，你会好受些……
>
> 摘自 Harry Nilsson 演唱的 Coconut 一歌中的歌词

高级大纲中的第 2 章是关于测试过程以及这个过程中发生的活动，它为大纲中对应概念包含的内容建立了一个框架，使读者能够对余下的概念原理有个感官认识。该章包括以下 7 个章节：

1. 概述。
2. 测试过程模型。
3. 测试计划和控制。
4. 测试分析和设计。
5. 测试实现和执行。
6. 评估出口准则和报告。
7. 测试结束活动。

下面介绍每节以及其与测试分析的关系。

2.1 概　　述

学习目标：

能回想起该部分内容即可。

ISTQB 基础级大纲中介绍了 ISTQB 基本的测试过程，该过程是一个通用的、可定制的过程，如图 2-1 所示。该过程由以下活动组成：

- 计划和控制；
- 分析和设计；
- 实现和执行；
- 评估出口准则和报告；
- 测试结束活动。

对于软件测试分析师而言，可以只把重点放在中间三项活动上。

pin51.com 最新最全资源分享 拼吾爱

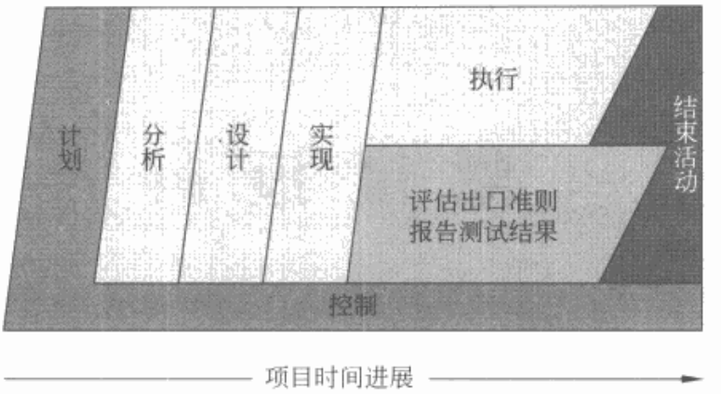


图 2-1 ISTQB 基本测试过程

2.2 测试过程模型

学习目标:

能回想起该部分内容即可。

本部分的概念主要适用于测试经理，对软件测试分析师是不做要求的。在针对认证考试时的学习过程中，只需阅读高级大纲第 2 章相关部分，对其熟悉，能回想起其中的内容即可。

2.3 测试计划和控制

学习目标:

能回想起该部分内容即可。

本部分的概念主要适用于测试经理，对测试分析师是不做要求的。在针对认证考试时的学习过程中，只需阅读高级大纲第 2 章相关部分，对其熟悉，能回想起其中的内容即可。

2.4 测试分析和设计

学习目标:

（K2）解释在软件应用生命周期特定阶段开展功能性测试的目的。
（K2）举例说明准则如何影响测试条件开发的结构和级别。
（K2）描述测试分析和设计如何成为发现缺陷的静态测试技术。
（K2）举例说明什么是测试准则，以及测试准则如何应用在测试规约说明中。

在测试过程中的测试计划活动中，测试负责人和测试经理将与项目的利益相关者一同制定测试目标。我们之前已经在基础级大纲中介绍过的电气电子工程师协会（IEEE）提出

的 IEEE 829 模板，会在本书之后的部分加以回顾。测试负责人或是经理可以在（IEEE 829 模板）“待测试特性”一节中找到相关模板将这些测试目标文档化。

对于测试分析师来说，测试目标是一个重要的产出物，没有了测试目标，将无法知道要测试什么。在测试分析和设计的活动中，需要用这些测试目标来指导以下两个主要子活动的执行：

- 识别每个测试目标的测试条件，并加以精炼。
- 生成检验测试条件的测试用例。

ISTQB 术语

测试策划：制定或更新测试计划的活动。

测试计划：描述预期测试活动的范围、方法、资源和进度的文档。它标识了测试项、需测试的特性、测试任务、任务负责人、测试员的独立程度、测试环境、测试设计技术、测试的进入和出口准则和选择的合理性、需要紧急预案的风险，是测试策划过程的一份记录。

测试用例：为特定目标或测试条件（例如，执行特定的程序路径，或是验证与特定需求的一致性）而制定的一组输入值、执行入口条件、预期结果和执行出口条件。

测试条件：组件或系统中能被一个或多个测试用例验证的条目或事件。例如，功能、事务、特性、质量属性或者结构化元素。

然而，仅测试目标本身是不够的，我们不仅需要知道要测试什么，还要知道是以什么顺序进行测试，测试到什么程度。由于时间的局限性，对最重要的部分优先测试的渴求，以及希望测试尽可能地最有效力和效率等需求，我们需要为测试条件划分出优先次序。

当采用基于风险的测试策略（我们将在第 3 章详细讨论）时，测试条件就是在质量风险分析中识别出来的质量风险项，那么优先级的设定取决于风险项发生的可能性和影响；也就是说，我们需要评估每个风险项的风险等级。优先级决定了（在整个测试过程中）测试工作量的分配，以及测试设计、实现和执行的顺序。

在整个测试过程中，已确定的测试条件及其优先级可以随着项目的（进展）需求或是我们对项目需求理解的深入加以调整，或者随项目利益相关者（群体）的演变而调整。

这种对（测试）优先次序的排序、应用和更新在测试过程中是很常见的，它起始于风险分析和测试计划阶段，贯穿整个测试过程，从分析和设计到实现和执行，并将影响对出口准则的评估和测试结果的记录。

ISTQB 术语

出口准则：和利益相关者达成一致的一系列通用和专门的条件，来正式地定义一个过程的结束点。出口准则的目的可以防止将没有完成的任务错误地看成任务已经完成。测试中使用的出口准则可以来报告和计划什么时候可以停止测试。

测试执行：对被测组件或系统执行测试，产生实际结果的过程。

pin51.com 拼吾爱 最新最全源码资源下载

2.4.1 功能测试目的

在更深入地讨论测试过程之前，让我们先来看看功能测试的目标。

首先，需要记住的重点是功能测试的测试目标适用于任何一个测试级别，并存在于软件生命周期的始终。测试团队往往在项目临近结束的时候才强调其主要的功能测试目标，以至于检测到不得不使项目停止的失效时，咬牙悲叹，追悔莫及。

以某个视频游戏为例。对于一个视频游戏，大多数用户可能会认为存储和重载游戏进度是其主要的功能之一，所以测试该特性就成为一个主要的测试目标。

对上述测试的明智做法是，项目组的各成员应该在单元、集成、系统和验收各个测试级别对这一功能进行测试。而如果等到系统测试阶段，才去检测存储、重载的缺陷恐怕会影响项目进度，那对于像游戏这样一个消费性产品是很不利的。

此外，为什么一定要等到各个测试级别的执行测试阶段开始才对该功能进行测试，早一些或者是晚一些呢？其实，通过在测试执行阶段前后的评审需求规格说明书、设计规格说明书和调用此功能的代码也同样可以对该功能进行评测。

另外我们可以利用现有的所有信息和资源，分析测试依据、测试目的、质量风险等识别测试条件，我将在第3章就这种基于风险分析的测试策略进行更详细的解释。

如果您不是使用基于风险分析的测试策略，那么则需要根据您采用的测试策略，选择一些具体的输入数据和技术。这些测试策略、输入数据和技术需要与测试计划保持一致，当然也要与任一具有广泛意义的测试方针或测试手册相一致。

本书的内容主要与软件测试分析师的这个角色相关，所以我们将重点放在功能测试上面。然而，我在这里需要指出的是，测试分析师需要具备识别功能和非功能测试条件的能力。我们也需要考虑到测试项目中被强调的功能和非功能特性的测试条件的详细程度和文档结构。

在识别和文档化测试条件时，有两个重要的抉择：

- 测试条件描述的详细程度；
- 测试条件的文档结构。

有许多种常用的方法可以用来决定测试条件的详细程度和结构。

其中方法是与描述测试依据的文档并行处理。例如，如果您公司有一份市场需求文档和一份系统需求文档，前者通常是概要性的，而后者是详细的。那么，您可以利用市场需求文档识别一些概要测试条件，然后再利用系统需求文档识别一些更详尽的在这些概要的测试条件之下的详细的测试条件。

另外一种方法通常与质量风险分析相关（有时叫做产品风险分析）。在这一方法中，我们可以概括出一些关键的系统或产品特性和质量特征，然后针对每一个特性或特征识别出一个或多个细化的质量风险项，那么通过这些细化的质量风险项便可以得出具体的测试条件。

再有另外一种方法，如果您只有比较详细的可直接撰写详细需求（规格说明书）的需求。这种情况下，需要保证从具体的测试条件到（会影响结构的）需求的可追溯性。无论是向管理层的报告，还是在对测试进行文档化时都需要这种可追溯性。

还有一种方法仅用来识别概要测试条件，有时并没有正式的测试依据。例如，在探索性测试（Exploratory test）中，一些人主张按照类似于测试宪章的形式文档化测试条件。那么，在这个阶段，对这种没有或者有极少脚本的测试几乎没有或者没有额外进行细化。

再强调一次，测试条件的详细程度和文档结构必须要与测试策略保持一致，而这些测试策略要和测试计划保持一致，当然也要与任一具有广泛意义的测试方针或测试手册相一致。

同时，请不要忘记，如果是从如需求、设计、用例、用户手册等这样的测试依据文档中得出测试条件的，就很容易保证信息的可追溯性。如果是这之后通过对测试用例的评审重现这种追溯性则难度很大。

让我们来看一个如何在此阶段（测试分析与设计）应用基于风险的测试策略识别测试条件的例子。

假设您正在测试一个银行在线系统。在风险分析时，一个关键的系统性能——系统响应时间，被认为是该银行在线系统的高风险领域。可能会有多个潜在的失效，而每个失效都有其相应的发生概率和影响。

所以，跟利益相关者的讨论引导我们仔细推敲系统性能的风险域，识别出 3 个质量风险项：

- 登录响应速度慢；
- 查询响应速度慢；
- 事件交换时响应速度慢。

到了这个阶段，细节的详细程度就足以令风险分析团队能够为每个风险项分配具体的发生概率和影响的优先级。

此时我们有了测试条件，下一步通常就要将这些条件细化到每个测试用例中。我说“通常”是因为，一些测试策略，如在基础级大纲和本书的第 4 章中讨论过的一些测试策略，例如互动策略，通常是不需要写测试用例的。在这里我们先假设我们需要开发的测试用例是可重复使用的、可验证的和可追溯回至需求、质量风险或其他任何测试依据。

ISTQB 术语

测试设计：将测试目标转换成具体的测试条件和测试用例的过程。

测试设计规格说明：为一个测试条目指定测试条件（覆盖项）、具体测试方法并识别相关高层测试用例的文档。

概要测试用例：没有具体的（实现级别）输入数据和预期结果的测试用例。实际值没有定义或是可变的，而用逻辑概念来代替。

详细测试用例：具有具体的（实现级别）输入数据和预期结果的测试用例。抽象测试用例中所使用的逻辑运算符被替换为对应于逻辑运算符作用的实际值。

在准备创建测试用例时，对于一个给定的测试条件、或者两个或以上的相关测试条件，可以应用不同的测试设计技术。这些技术将在第 4 章讲到，记住，您能够也应该做的，是将多个测试技术融入到一个测试用例的创建中。

pin51.com 最新拼吾爱

上面提到过的关于需求、质量风险或任何测试依据的可追溯性，我们可以通过直接关联测试用例到测试依据，或者到已经创建的测试用例引发出的测试条件的原始依据来建立追溯关系；或者我们可以通过间接地关联测试用例和测试条件，再关联到依次产生这些测试条件的测试依据来建立追溯关系。

与测试条件一样，我们需要为我们的测试用例选择详细程度和文档结构，选定的详细程度和文档结构必须要与测试策略保持一致，而这些测试策略要和测试计划保持一致，当然也要与任一具有广泛意义的测试方针或测试手册相一致。

这时，还有什么与测试设计过程相关呢？虽然测试设计的具体过程取决于选用的具体技术，但测试设计普遍包含对以下各项的定义：

- 前置条件；
- 测试环境需求；
- 测试输入数据和其他测试数据需求；
- 预期结果；
- 后置条件。

ISTQB 术语

测试实现：开发、排序测试规程，创建测试数据，若需要，还包括准备测试用具和编写自动化测试脚本的过程。

定义测试的预期结果就有点难度了，尤其是预期结果不仅仅是显示在屏幕上的输出，还有数据和环境后置条件，解决这一问题，要求使用一种叫做测试准则（test oracle）的技术，等一下我们就会讨论到这个问题。

首先提一下对测试环境的需求，这是在 ISTQB 基础级测试过程中比较模糊的一个部分，确切地说，到底什么是测试设计阶段和测试实现阶段的分界线？

高级大纲中指出，“在测试设计阶段，需要定义出测试基础设施的具体需求，尽管在实际情况下对这些需求的定义要一直到测试实现阶段才有可能最终完成”，这样来说，或许设计阶段我们就对某些工作进行了具体实现？两者不可以重叠吗？对于我来讲，在许多诸如此类的问题中纠结于如何区分其差别，无异于回答“多少个天使能在一个针尖上跳舞”一样，没有实际意义。

无论人们如何定义测试环境和基础设施——设计、实现、环境搭建还是其他别的名词，重要的是要知道测试包含的不仅仅有测试目标和测试件，还有测试环境，而且测试环境不仅仅只有硬件，还包括房间、设备、人力、软件、工具、外围设备、通讯设备、用户授权和其他所有运行测试需要的物品。

2.4.2 测试准则

下面介绍什么是测试准则，以及测试分析师可能会面临的与测试准则相关的问题。

可根据测试准则用以决定测试预期结果的源。测试中，我们可以将实际运行的结果与这些预期的结果相比较。有时候准则是一套现成的系统，有时候是一份用户操作手册，

有时候是某个个人的专业知识。我们不应该将代码本身作为测试准则，即使是作为结构测试的测试准则，否则就变成了纯粹对编译器、操作系统和硬件的测试。

所以，什么才是测试准则相关的问题呢？如果您没有一些处理这类问题的经验，那么先问自己一个问题，一般来说，我们是如何知道什么是测试的“正确结果”？对其认知的困难就是一个测试准则相关问题。

如果您刚刚离开大学这座象牙塔，进入职场，也许您已经学习过一些完美的软件工程项目的范本。您恐怕也听说了一些这样的故事，这些项目具备详细的、明晰的和一致的测试依据，例如需求和设计规格说明书，并在其中对所有的预期结果做出了定义。如果您真的听说了，那么我需要告诉您，这些故事都只是神话。

在现实中，一个真正的项目，测试依据文档例如需求文档通常是模糊不清的，两份文档，像是市场需求说明书和系统需求说明书，经常都是互相矛盾的。这些文档往往有漏洞，且忽略对产品的重要特性的论述，尤其是非功能特性、特别是易用性和用户界面的特性。

有时候这些文档干脆就没有，有时候即使有却非常肤浅，几乎没什么用。我的一个客户曾给我看过一份在一张信封大小的纸片上随手写的文稿，上面还有些粗糙的示意图，这居然就是一个要花费 100 个人月左右的项目的全部需求说明。

等到测试依据文档已交付时，而通常会交付得比较晚，晚到在因无法继续等待而不得不开始测试设计（如果还想在测试开始前完成测试设计的话）。

即便是业务分析师、销售和市场部员工，还有用户都做得很好，尽到了他们最大的努力，测试依据文档也不可能完美。而现实中的应用程序如此复杂，也不可能完全被修改（记录）成完整、明确的测试依据规格说明文档。

因此，对于收到的写好的测试依据文档，不得不以测试专业的判断和职业悲观主义思想，自己或者与测试专家一起再次对其考证。通过使用各种已有的准则，无论是书面的和脑海中的，还是别人提供的和推导出来的，测试工程师（这里指测试分析师）可以在测试执行之前或期间定义预期结果。

因为前面已经讨论过很多关于需求的内容，您可能会认为测试准则相关问题仅适用于概要测试，例如系统测试和验收测试。其实不然，测试准则相关问题及其解决办法，适用于所有的测试级别。概要测试例如用户验收测试和系统测试更依赖于需求规格说明书、用例和预定义的业务流程，详细测试例如组件测试和集成测试则更依赖于详细设计规格说明书。

虽然存在争议，但是您应该牢记必须要在测试过程中解决测试准则相关问题。测试准则出现问题导致无法评估其结果的测试就是浪费时间。这样的测试为软件项目团队提供的价值是极低的或者根本没有价值，甚至会造成负面影响。这样的测试产生的假象测试结果会分散软件项目团队的精力，也会造成对系统质量信心认知的错误。

顺便提一句，关于刚刚提到的那个“象牙塔”，当我很多年前在加州大学洛杉矶分校（UCLA）学习计算机科学的时候，我的软件工程教授从一开始就告诉过我这个关于准则的问题，所以我想我不能说我没被提醒过。

下面是现实中关于测试准则的一个例子。

pin51.com 最新最全 拼吾爱

我曾经参与过一个银行的项目，开发一个应用程序取代以前的遗留系统，当时有两个测试准则，一个是基本保持了其原始状态的需求规格说明书，另一个是遗留系统本身。于是，我们需要面临两个挑战。

第一个挑战是，需求很不明晰，供应商对项目的最初理解是“客户需要什么就提供什么”，接着他们意识到，客户及其终端用户对系统应该具备的功能的期望是毫无节制的和相互冲突的， 如果完全按照这些意愿加以开发的话，他们很快会陷入破产的境地。我们目前得到的这份需求文档是一份发现上述问题，再调整结构而迟到的产出物。

第二个挑战是，有时新系统和遗留系统的差别并不是很大，我记得有这样一个声名狼藉的情况，曾经有一份关于某个问题的测试缺陷报告，我们打开、推延，再打开、再推延，这样来来回回至少有四五次，而该测试报告描述的只是关于每月有 0.01 美元变化的付款问题。

缺乏一组可靠的、权威的、一致的测试准则将导致一种“缺陷报告的乒乓球”现象，同时我们也有可能面临测试缺陷报告的优先级问题，因为人们总是在争论一些问题是不是真的是问题，从而出现很高的漏报和误报率。而对于整个软件项目团队，包括测试团队，将会因此很泄气。测试准则不仅仅是一个抽象的概念，而是现实需求产生的系列结果。

2.4.3 标准

在基础级大纲中就曾提到过的对测试分析和设计有效的标准。

先来看两个文档模板，它们可以帮助您将在分析和设计测试时捕获的信息文档化。第一份是 IEEE 829 测试设计规格说明。

测试设计规格说明是用来描述测试用例及其概要测试条件的。该模板包含以下几个部分：

- 测试设计规格标识（遵循您公司现行的文件标识标准即可）；
- 待测试特性（在测试套件中）；
- 详细的方法说明（已经确定了的要采用的技术、工具等）；
- 测试标识（以追踪测试套件中的测试用例）；
- 特性的入口/出口准则（例如，我们打算如何确定某个特性是正常运行的，例如，是通过测试准则、测试依据文档，还是遗留系统）。

在测试设计规格说明中若干测试用例的集合通常被称为测试套件。

测试套件和测试套件中测试用例的执行顺序往往是由风险和业务的优先级驱动的。当然，项目的局限性、资源及过程也肯定会影响到测试套件的执行顺序。

下面介绍 IEEE 829 中的测试用例规格说明。测试用例规格说明中会描述测试用例的细节。该模板中包括以下几个部分：

- 测试用例规格标识；
- 测试项（哪些是待交付的和待测试的）；
- 输入规格说明（用户输入数据、文档等）；
- 输出规格说明（预期结果，包括屏幕输出、文档、时间等）；

pin51.com 最新最全资源分享

- 环境需求（硬件、软件、人员、道具等）；
- 特殊的程序需求（技术人员的介入、许可等）；
- 用例之间的依赖关系（如果有必要的话、需要设定预置条件）。

然而这个模板仅仅定义了如何描述测试用例内容的标准，而没有涵盖如何界定该测试用例是否设计合理或是否值得成为一个测试用例。实际上，不同测试用例的工作量、持续时间和所包含测试条件的数量是有显著不同的。

在下一章节将讨论 IEEE 829 标准。现在讨论一个在基础级大纲就涉及的关于文档化的问题。

在现实世界中，测试文档化的程度呈现了很大的多样性。关于其变化的原因很难一一列举，不过可以概括如下：

- 有无文档化对项目的风险。
- 测试文档可能或应该会创造多大的价值。
- 已经有或者应该遵循的标准。
- 项目采用的软件开发生命周期模型。敏捷开发方法的提倡者试图通过团队成员间密切而频繁的交流将文档最少化。
- 需要我们提供从测试依据到测试用例的可追溯性的详尽程度。

关键的一点是，您需要在决定测试文档化的程度时保持开明的思想和清醒的头脑。

正如我之前说过的，我们需要侧重于测试分析师教程中的功能特性，现在让我们一起回顾 ISO 9126 标准。

ISO 9126 质量标准定义了 6 个软件的质量特性：功能性、可靠性、易用性、效率、可维护性和可移植性。每个特性都有 3 个以上的子特性，如图 2-2 所示。

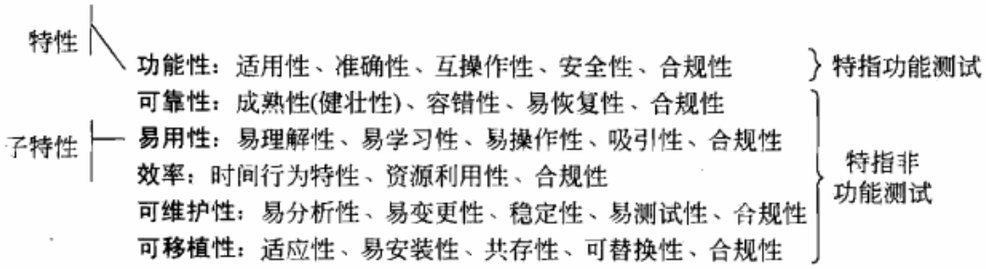


图 2-2 ISO 9126 质量标准

针对功能性及其子特性的测试就是功能测试，是本书的侧重点，而测试范围将决定需要对哪些具体的子特性进行功能测试。

针对其他 5 个特性及其子特性的测试就是非功能测试。将在测试技术分析员系列书籍的另外一册中广泛涉及。最后，您需要记住的是，在测试硬件/软件系统时，还可能包括针对一些额外的质量特性的测试。

2.4.4 静态测试

下面回顾一下在基础级大纲中提到的 3 个重要观点。其一，在软件生命周期中及早进行静态测试的价值所在，即此时捕捉和修正缺陷的成本都会很低；其二接下来是，在

pin51.com 最新最全 拼吾爱

软件生命周期的早期就引入测试可起到防御功效；其三，测试应该在项目的早期就参与进来。这 3 个观点相互关联因为测试分析和设计本身就是一种静态测试形式，该形式与其他静态测试形式相互协作，而这种协同性，只有这些静态测试在恰当的时机相互配合，其价值才能被发掘出来。

要注意，仅在测试分析和设计完成时，您才有可能在对测试依据进行评审和静态分析的同时，对测试条件和测试用例进行定义。事实上，您可以就需求的测试分析和设计准备一个需求评审会议。测试分析和设计可以视为一个针对需求规格的、体系化的、关注失效的静态测试，从而为需求评审会议提供有价值的输入。

当然我们需要利用静态测试的优势，尽早使得测试和非测试的相关人员参与到各个测试产物，包括风险分析、测试设计、测试用例和测试计划等的评审工作当中。我们还需要将适当的静态分析技术运用到这些测试产物中。

让我们来看一个测试分析是如何起到静态测试作用的例子。假设您是按照基于风险的测试策略进行测试，除了各个质量风险项可以作为测试条件之外，一个具代表性的质量风险分析环节还可以产出另外一些有效的产物。

可以称这些产物为次级产物，像副产品生产线一样，这些次级产物是在创造目标产物时产生的，在这个例子中，目标产物就是一份质量风险分析文档。这些次级产物是由于您和其他参与者在质量风险分析过程中对项目未曾考虑的（风险）部分加以注意而产生的。

这些次级产物包括：

- 项目风险——可能发生并威胁到项目的成功；
- 识别需求规格说明、设计规格说明或者是质量风险分析中使用到的其他文档中的缺陷；
- 列出对（测试）实现的假设和简化，不仅可以改进设计，同时也可以设定一些检查点以确保风险分析能够被融合到之后的测试实现中去。

通过将这些次级产物交付给相关的项目组成员，可以预防缺陷流入软件生命周期的后续阶段，这终归是一件好事。

> **ISTQB 术语**
>
> **测试规程**：规定了执行测试的一系列行为的文档。也称为测试脚本或手工测试脚本。
>
> **测试脚本**：通常指测试规程规约，尤其是自动化的。

2.4.5 度量元

在本节的结尾部分，让我们来看看对于测试分析和设计的度量元和度量。对于测试分析和设计这部分的完整性，可以通过以下几个方面进行测量：

- 测试条件对需求或者质量（产品）风险的覆盖率；
- 测试用例对测试条件的覆盖率；

■ 在测试分析和设计过程中发现的缺陷的数量。

一个很有效的方法是，可以通过对测试分析和设计活动各项工作的分解内容的追踪和测量，以决定我们的测试是否是按照预计和进度进行的。

2.5 测试实现和执行

学习目标:

（K2）描述测试执行的前置条件，包括测试件、测试环境、配置管理和缺陷管理。

测试实现包括所有能驱动测试执行开始的余下的测试任务。这时，我们已经完成分析和设计的工作，那么余下的指什么呢？

首先，如果我们想采用明确的、具体的测试规程，那么需要将测试用例组织成测试规程或测试脚本，而非依靠测试员的经验，我所说的，“组织测试用例”是指，至少要写出测试的执行步骤。但是这个步骤要多详细呢？这应该与对测试条件和测试用例级别的详细程度的考虑和要求相同，例如，如果应用一个类似于美国联邦航空局（United States Federal Aviation）DO-178B 的标准，就需要达到比较详细的级别。

在测试中，经常需要测试数据，包括各种输入和测试环境本身，需要确保这些输入数据在此阶段已经准备好，同时，还必须搭建测试环境。测试前，需要先确认一个问题，就是测试数据和测试环境已经准备到最佳状态以开始测试了吗？如果答案是“不”，那么我们必须在测试执行开始之前把这些问题解决掉。某些情况下测试数据需要借助数据生成工具或者生产数据（如银行系统）；而为了确保测试环境配置的正确性，也可以使用配置管理工具。

有了测试规程后，就需要将这些测试规程合并起来，制作一份测试执行的进度安排。谁来执行测试？以什么样的次序来执行？需要哪些测试环境？这些问题都需要被一一解决。

最后，因为测试执行即将开始，我们需要检查是否已经满足了所有的显性或是隐性的入口准则。如果尚未满足，则需要和项目的利益相关人员通力协作，保证问题能在预定的测试执行开始日期之前解决。

请记住，您应该为测试规程排出优先级和进度安排，以确保能用高效的方式达成测试策略中的目标。以基于风险的测试为例，我们通常就按照风险的优先级顺序执行相应的测试用例。当然现实中还存在很多局限性，例如测试配置的可用性很可能会影响到测试执行的顺序，出于效率的考虑，例如在一项测试结束之后，像对测试数据量和测试环境复位等的效能方面的考虑，也可能会改变测试的顺序。

下面进一步讨论两个重要的内容，测试规程和测试环境的准备工作。

2.5.1 测试规程的准备工作

这个测试规程准备好被执行了吗？在回答这个问题之前先检查几个有必要强调的问题。

pin51.com 最新最全源码分享 拼吾爱

正如前面所提到的，必须明确测试规程中的一系列问题，包括识别由谁来执行测试，什么时候，在什么样的测试环境，用哪些测试数据。

必须评估现实中的局限性，很有可能因为这一局限性，测试需要按照特定的顺序执行。假设我们有一组测试规程，这样就有可能组成一个端对端的工作流吗？可能还需要遵守可能存在的能够决定这些测试规程运行次序的业务规则。

所以，综合理论上的理想测试规程的顺序以及实际情况的考虑（从最重要的到最次要的），需要将测试规程的次序最终确定下来，这包括与测试团队和其他利益相关成员确认这一次序。在确定该次序的过程中，您可能会发现您认为可能遵从的顺序事实上是不可操作的，或是效率低到无法被接受。

很可能还需要采取一些措施实现测试的自动化。当然，很可能不是必须，因为不是所有的测试都需要引入自动化的。如果一些测试是自动的，必须确定这些测试应如何适合于测试次序。如果与手动测试在相同的（测试）环境下进行自动化测试，会很容易损坏或破坏测试数据，有时会导致手动测试和自动化测试都产生大量的缺陷漏报和误报。猜猜看后果是什么？这就意味着您需要重新运行一遍所有的测试，这肯定是我们不想看到的。

高级大纲中说我们需要在测试实现阶段创建测试用具和测试脚本，理论上来讲是正确的，但是实际情况是，您确实需要在执行测试脚本的数月前，最少也是数周前将使脚本自动化的测试用具准备好。

肯定需要了解所有测试规程之间的依赖关系，如果发现由于这些依赖关系无法按照之前计划的顺序执行这些测试规程，那么有两个选择，一个是可以更改顺序以适应已发现的各种障碍，另一个就是可以消除这些障碍。

下面介绍两类比较普遍的测试规程的依赖关系的分类以及因此而引发的障碍。

首先，第一类依赖关系是针对测试环境的。您需要清楚每个测试规程对测试环境的要求是什么，请检查一下，在运行预先安排好的测试规程期间，其测试环境是否可用。请注意“可用”在这里不仅指测试环境已配置好，还指没有其他的测试规程或是因为其他相关测试活动，在同一时期也在为该测试规程事先安排好的测试环境中进行而产生干扰。

干扰问题通常就是障碍出现的地方。对于一个复杂的和不断演进的测试环境，单纯对测试环境的配置管理将会成为一个问题。一段时间以前，我曾经参与了一个项目，由于这个项目比较复杂，不得不专门创建了一个特殊的数据库用来追踪、记录和管理测试规程和它们所需的测试环境之间的关系。

测试规程的第二类依赖关系来自测试数据。您需要知道每个测试规程需要什么样的数据，与之前的流程相似，需要先检查在您运行预先安排好的测试规程期间，测试数据是否可用。和之前说过的一样，“可用”不仅指测试数据已创建，还指没有其他测试数据或是其他相关的测试活动，在同一时期也调用或衍生事先为该测试规程创建好的测试数据而产生干扰。

在测试数据方面，干扰通常是个大问题。我曾经有个客户试图在白天进行手动测试，通宵进行自动测试，引发了很多的问题。直到改善了流程——在手动测试和自动测试的交接点，也就是白天的开始或结束前，妥善恢复各自的测试数据，这一问题才得以解决。

pin51.com 最新最全源分享下载

2.5.2 测试环境的准备工作

可以在这个测试环境中进行测试了吗？在回答这个问题之前先检查几个有必要强调的问题。

首先，需要搞清楚，正确配置测试环境的重要性。如果是在一个配置错误的测试环境下执行测试规程，无论多么完美地运行了这些测试规程，只能得到无用的测试结果。特别是，可能会得到很多误报。在软件测试中一个误报就好比在医学中的一个误诊，即一个测试实验本应该是通过的，结果误判为错误，导致人们浪费很多时间分析这些“缺陷”，结果其实是测试环境的问题。通常在这种情况下误报数量会非常巨大，以致我们也会得到漏报，也就是说一项测试本来不应该判为通过的，却通过了，因为我们无法看到这些隐藏在（大量）误报中的缺陷。无论是误报还是漏报都会导致极低的缺陷检测率，极高的生产失效率和现场的失效率，导致很高的缺陷报告的被拒率，浪费测试工程师、经理和开发工程师的时间，也将会导致整个测试团队信用的严重丧失。还需要列举更多的损害吗？

所以，我们必须确保测试环境的正确配置。在 ISTQB 的基本测试过程中，测试环境配置是在测试实现阶段进行的。但是如果是自动化测试，即使我们在测试分析和设计完成之后就开始测试实现，也可能已经太迟了。如果我们把测试环境的实现视为整个测试实现阶段活动中的一个子集，在测试计划刚结束的时候就开始进行，那我们的处境就会好很多。

什么才是正确配置的测试环境呢？我们能从中得到什么益处呢？

首先，一个正确配置的测试环境使我们有可能在企图运行的测试条件下发现缺陷。例如，如果我们计划进行性能测试，正确配置的测试环境使得我们能发现导致系统变慢的非预期的瓶颈。

其次，正确配置的测试环境能够确保当缺陷尚未发生时能正常运行，也就说不会产生很多误报。

此外，对于概要测试，例如系统测试和系统集成测试，正确配置的测试环境应该是对生产或是终端客户环境的复制。很多缺陷，尤其是非功能性缺陷如性能和可靠性的缺陷，如果在缩减了规模的环境中无法找到的话，那么就麻烦了。

对于正确配置的测试环境，还需要有人搭建和支持这个环境。对于复杂的测试环境，这项工作通常是由非测试团队的人员来承担。我们还需要确保这个人（无论是测试工程师，还是其他人员），已经将测试件、测试支持工具和相关的过程加载到测试环境中了。

测试支持工具至少包括配置管理、时间管理、测试日志备份和测试管理。至少您需要为出口准则评估和测试结果报告收集数据的规程（或流程）。理想状况下，测试管理系统应该会帮您完成其中的一些工作。

ISTQB 词汇

测试日志：按照时间顺序排序的有关测试执行所有相关细节的记录。

测试记录：将测试执行信息写进测试日志的过程。

pin51.com 最新最全源分享下载 拼吾爱

2.5.3 混合型测试策略

混合型测试策略常常是个不错的想法，能在整个测试过程中，包括测试实现阶段，使测试方法得以平衡。比如说，当我和我的助手一起做项目时，我们一向都将基于风险分析的测试策略和动态测试策略混合在一起，预留一定比例（通常是 10%～20%）的测试执行工作量，用来进行那些没有使用预先定义脚本的测试。

分析性测试策略恰好符合 ISTQB 中的基本测试过程，并在这一过程中产出相应的交付物。然而，对混合型策略带来的风险主要在于动态反应测试的部分可能会失控。没有脚本的测试，不应该是混乱的或是漫无目的的，但这些测试的持续时间和覆盖率是很难预测的。

一些测试技术，例如基于会话的测试管理技术，可以在一些反应式策略中帮助我们解决一些内在控制的问题，我们会在稍后讲解该技术。此外，对于构造反应式测试策略，可以使用基于经验的测试技术，例如有计划的攻击、错误猜测和探索性测试。我们会在第 4 章讨论这些问题。

反应式测试策略的普遍特征是，大部分测试是针对呈现在我们面前的真实系统的实际表现做出反应。也就是意味着测试分析、测试设计和测试实现主要发生在测试执行阶段。换言之，反应式测试策略允许，事实上是需要每个测试的结果为后续测试的分析、设计和实现提供依据。

正如在基础级大纲中讨论过的，无论是在测试执行之前还是测试执行期间，这些反应式测试策略在整个测试中所占的比例是较低的。基于经验的测试技术通常是找到漏洞的有效的方式，效率有时甚至是脚本测试技术的 5 或 10 倍。当然，这种基于经验的测试策略需要由测试专家来运用。正如我之前提到过的，反应式测试策略有时会导致测试执行的持续时间难以预测。这些测试策略的轻量属性（在整个测试中的比例低）意味着，它们不能提供太多覆盖率，也很难在回归测试中被重复利用。有人称工具可协助解决（反应式测试策略）关于覆盖率和可复用性的问题，不过在实际工作中，我还没有看到过有效的方案。

也就是说，当反应式测试策略和分析性测试策略相混合，它们可以互补彼此的弱点。就好比是一杯苏格兰威士忌的调和酒，这种调和酒是由麦芽鸡尾酒（不管是一种麦芽还是多种麦芽）混合在一起的威士忌，再掺入谷类酒（基本上都是伏特加酒）调和成的。很难想象出其他两种酒比伏特加和谷类酒之间的差异更大，但是当这两种酒调和在一起就变成了很易入口，比纯粹的烈性酒更为大众所喜爱的酒。

2.5.4 测试执行的开始

现在到了准备开始测试执行的时候了。在开始前，需要确认测试对象已经交付，并且满足测试入口准则（也有可能根据实际情况放弃了其中的一部分）。

当然，只是在假设仅满足入口准则就足以保证之前所讲过的测试实现任务已全部完成。如果该假设不成立的话，还需要回头再去检查一下那些关于测试数据、测试环境和测试规程依赖关系等的问题是否已经解决。

在测试执行阶段，人们通常按照测试规程，手动执行测试用例。一个测试规程从开始执行到完毕，应该至少完成了以下两件事情：首先是覆盖了所有的该测试规程可追溯到的源测试条件，或源质量风险项，其次，已经执行了该测试规程中的所有步骤。

您可能会问："怎么可能执行完了测试规程却没有覆盖所有的风险项和测试条件呢？"。有时候，测试规程只描述了一些概要层面的东西。这种情况下，您应该去理解该测试的含义，并根据当时的具体情况对已经写好的（概要）测试规程进行增补以确保恰好覆盖了其对应的（风险）领域。

您还有可能问："怎么可能覆盖了所有的风险项和测试条件而没执行完所有的测试规程呢？"一个测试规程中，某些步骤通常是确保测试能够正常进行，而不是用来为覆盖测试条件和风险项而刻意设置的。例如，有些步骤是创建测试数据或是其他的前置条件，有些步骤是收集日志信息，而有些步骤是用来在最后将系统复位到一个已知正确的状态。

第三类适用于手动测试执行期间的活动，可以在一定程度上把探索性测试合并到测试规程中。为了实现这一目的，有几种途径，一种是，我们可以不把测试规程定义得那么清晰，让测试工程师选择一种他们喜欢的方式来完成某一任务。另外一种方法是我通常采用的，就是告诉测试工程师测试脚本只是一个指示说明，哪些是需要注意的地方，当到了这部分时，他们需要停下来思考，这就允许他们超越、跳出脚本的框架，我发现这是一个很有效的方法。

最后，在测试执行过程中，工具通常被用来运行自动化测试，这些工具毫无偏差地按照之前定义好的脚本执行测试。最初看起来纯粹是"不错的事情"，然而，如果我们不能正确地设计测试脚本，那么就意味着脚本无法与正在测试的系统同步，导致了很多的误报。您可以阅读高级测试技术分析一书，里面有很多关于这一问题以及是如何解决这个问题的内容。

2.5.5　一个独立的测试规程的运行

让我们聚焦来看看一个测试工程师在运行一个独立的测试规程过程中的行为。在初始筹备中的后勤问题（如测试环境、数据等）处理好之后，测试工程师开始运行测试中的具体步骤，并产生实际的测试结果。

下面介绍测试执行的核心部分。可将实际的结果和预期的结果进行比较，这的确是检验测试到底增加了还是损害了整个项目团队价值的时刻。此前做的所有的测试设计和实现，都是为这一时刻准备的。此后的所有测试活动就是对此时比较结果的应用。因为这个环节对于良好的测试是如此关键，也是其核心部分，您应该非常关注并重视它。所以，如果我们发现实际结果和预期结果不相匹配怎么办呢？在 ISTQB 的术语中指出，每个实际结果和预期结果之间的差异都称为异常。一个不相匹配可以是多个差异，也可以说是多个异常。当我们观察到一个异常时，这就是一个事件。

有些事件是失效。失效指由于一个或多个缺陷导致系统不能正常运行，但这只是事件发生的一种理想状态。当我们发现失效时（真正由缺陷引发的），需要开始收集数据帮助开发人员解决这一缺陷。第 7 章会讲到更多关于事件报告和管理的细节。

一些事件并非失效，而是误报。误报指由于错误的测试规格、无效的数据、错误配

pin51.com 最新最全源码资源分享 拼吾爱

置的测试环境，或是在测试运行过程中，人为因素导致的错误，引发了实际结果和预期结果不相匹配。

如果能在误报发生时，及时发现是误报，就可以挽回一定的损失。测试员应该修正测试，如果测试需要登记到版本库（check into a repository）中，则有可能还需要一些配置管理的工作。测试员在这之后应该立刻重新运行该测试。这样，损失就可以仅仅控制在测试员个人浪费的时间范围内，也就是其可能影响测试进度的时间，加上修正测试的时间，再加重新运行该测试的时间。

如果测试员臆断该失效真实并对其进行报告，还是会衍生上述的所有活动，时间的损失以及对整体进度的延误。而且会浪费测试经理、测试工程师、项目开发等人员更多的时间，那么将更加得不偿失。

这里有一个关于误报的警言：如果一个测试在之前的多次运行中从未产生误报，这并不代表您可以对此次运行掉以轻心。测试依据、正确预期结果、测试对象等的变更都会使测试规格说明产生障碍或者无效（而导致原本正常的测试产生误报）。

2.5.6 测试结果的记录

很多测试工程师喜欢运行测试，至少是最开始的几次运行，但是他们并不是很喜欢记录测试结果，他们对这种文书工作通常是嗤之以鼻的，他们抗议说这是“官僚主义和繁文缛节”。

如果您是其中的一员，那么请摒弃这个观点。前面已提及所有的计划、分析、设计和实现的工作都是为了执行测试规程和比较实际和预期结果的这一刻准备的。我还说过，这一时刻过后的所有工作都是对比较的结果应用。那么，不去捕获这些结果的数值，它们就无法得以利用，而测试日志正是用以捕获这些数值的。

所以，请记住，测试员不仅要运行测试，还需要记录测试的结果。未能记录测试结果意味着，多半要重复测试，或是丢失了运行测试的结果。当您重做测试时，这是纯粹的浪费，浪费您运行测试的时间。因为测试执行通常是完成一个项目至关重要的部分，这一浪费，使得计划的项目结束日期存在风险，人们是很不希望看到这种情况发生的。

在我们讨论下一个话题之前，这里有一个旁注：我之前提过反应式测试策略及其存在的覆盖率的问题。请注意，有了足够的测试日志，当您无法提前确定反应式测试的覆盖率，但至少您可以在这之后得到关于覆盖率的信息。所以，再次强调一下，无论是否利用脚本测试，都请记录您的测试结果。

在测试执行期间，存在着很多变数。测试用例可能会有变更，测试对象和每个测试项组成部分也经常会有变化；测试环境、测试依据也可能会有所变动。所以测试日志中应注明测试时的版本（涉及到所有元素）。

军事战略专家克劳斯维茨（Clausewitz）曾提出著名的“战争之雾”理论，他这里所指的“雾”并非真正的雾——虽然在他那个年代，装着黑火药的大炮和众多轻武器制造出了很多雾——但是这里是借雾比喻无论您是步兵还是将军，没有人能看透一场战争的全局。

如果克劳斯维茨能够复活，并成为一名测试工程师，他可能会意识到测试工作中面

临的境况与他著名的“战争之雾”理论是如此相似。测试执行期间很容易存在大量的“雾”，那么好的测试日志就是一个驱雾器，测试日志应该为测试执行提供详细的内容和丰富的测试记录。

要做到这一点，测试日志需要按照每个测试、每个事件依次记录。每个测试都有唯一的标识，需要有在测试执行期间的状态信息的相关记录。这些信息不仅能帮助掌握测试进行的全局状态，还能帮助了解测试的整体覆盖率。

您还应该对测试执行期间发生的影响测试执行过程的重大事件加以记录，无论这些事件对测试产生的影响是直接的或是间接的，我们应该将所有延迟、中断或是阻碍测试的所有事件记录在文档中。

测试分析师并不总是测试经理，如不是就需要和测试经理密切合作。测试经理需要记录测试控制，测试进展报告和测试过程改善等的信息。测试分析师需要同测试经理一起记录用于出口准则的测度相关的信息，下一章节会讲到这一内容。

最后需要指出的是，测试日志的程度、形式和细节会根据测试级别、测试策略、测试工具和各种标准、规章的不同而异。自动组件测试的结果会自动地被记录下来；而手动验收测试通常需要测试经理根据测试分析师提供的信息，编写或至少要汇总相关的测试日志。如果我们测试像医药系统这样规范的安全关键系统，我们很可能需要为了审计的目的而记录一些具体的信息。

2.5.7 利用业余测试员

本章节是个相当令人振奋的部分，让我来解释一下业余测试员的含义：一个将测试当作主业，以测试谋生的人被称为专业测试员。其他所有参与到测试过程中的人都可以称之为业余测试员。

从 1987 年至今，我一直是一名专业测试员。这之前，我是一个专业程序员。我现在还会时不时地写一些程序，但是我现在只能算作一个业余的程序员，我经常会犯一些业余程序员普遍会犯的错误。在我成为专业测试员之前，作为程序员的我经常会对我写的代码进行一些单元测试，那时，我经常会犯一些业余测试员普遍会犯的错误。由于我做程序员时所在的那家公司的单元测试完全依赖于程序员，有时我所犯的业余测试员的错误会给客户和我带来难堪。

业余测试员的参与是值得鼓励的。有时我们希望用户或客户在测试执行期间充当业余测试员的角色。但是了解这一行为的目的以及为什么这样做会（或不会）行得通是很重要的。举例来讲，这一行为的目的常常是为了建立用户对系统的信心，但是常常适得其反！如果让他们参与得太早，这时系统还存在很多缺陷，那就比较狼狈了！

2.5.8 标准

下面讨论与测试实现和测试执行以及其他测试活动相关的标准。

首先从 IEEE 829 这一标准开始，关于 IEEE 829 的大部分内容在您看来可能只是对基础级大纲的一个回顾，但是经过深入研究之后，您就会发现也不尽然。

在测试实现阶段使用的第一个 IEEE 829 模板就是 IEEE 829 测试规程说明。该测试

pin51.com 最新最全源码资料下载 拼吾爱

规程说明描述了如何运行一个或者多个测试用例。该模板包括以下几个部分：

- 测试规程规格标识；
- 目的（例如，运行什么测试）；
- 特定要求（技能、授权许可、环境等）；
- 程序的步骤，包括记录、准备工作、开始、进展（就是步骤本身）、测量结果、关闭/挂起、重新启动（如果需要的话）、停止、封装/拆卸和突发事件。

虽然 IEEE 829 标准会对测试规程和测试用例加以区分，但是事实上测试规程常常会嵌入在测试用例中。

一个测试规程有时被称作测试脚本。测试脚本可以是手动的或是自动的。

IEEE 829 测试文档标准也提供了一些关于测试日志中应该包含哪些内容的建议。根据该标准，测试日志应该记录测试执行中的相关细节。关于测试日志的模板包含以下几个部分：

- 测试日志标识；
- 对测试包括各测试项（包括版本号）、运行的测试环境等类似问题的描述；
- 对活动和事件的输入，需要按照各个测试，各个事件一一罗列出来。事件包括测试环境无法使用，有相关人员生病无法工作等。您需要在测试执行过程中捕获以下这些信息：测试结果；环境的变更或其中出现的问题；观测到的缺陷、事件或是异常的现象；参与的测试员；任何测试的悬挂或是积压；计划的变更和变更的影响等。

英国标准机构制定了 BS 7925/2 标准。主要包括两大部分：测试设计技术和测试测度技术。

对于测试设计，该标准对测试技术进行了广泛的讨论，包括：黑盒测试、白盒测试等技术。该标准包含了基础级大纲中也涵盖的下列黑盒测试：

- 等价类划分测试；
- 边界值分析测试；
- 状态转换测试。

该标准还包括一种叫做因果图法的黑盒测试技术，该技术是决策表的图形化表达；除了因果图法，还有一种叫做语法测试。

该标准还包含了以下几种在基础级大纲中提到过的白盒测试技术：

- 语句测试；
- 分支和判定测试。

它还包含了其他几种基础级大纲中简要甚至没有提到的白盒测试技术：

- 数据流测试；
- 分支条件测试；
- 分支条件组合测试；
- 修正条件判定测试；
- 线性代码顺序和跳转（LCSAJ）测试。

该标准还包括“随机测试”和“其他测试技术”。基础级大纲中并没有提到过关于随

机测试内容，本书也不会包含该部分内容，不过会在关于高级测试技术分析那本书中讲到。该标准中的其他测试技术没有提供任何举例，只是讲了如何选择这些技术的规则。

这时您可能会说“等一下，讲得太快了，哪些才是在高级软件测试分析师认证考试中掌握的呢？”答案分为两部分，首先，最好能掌握所有基础级大纲中包含的测试设计技术。这部分内容肯定会对高级软件测试分析师认证考试有效。其次，认证考试的范围还会包含本书第 4 章讲到的一些新的测试设计技术。

BS 7925/2 为每个测试的度量技术提供了一个或多个覆盖率度量元。但由于没有在设计测试技术的同时解释覆盖率度量元的设计，根据该标准构建度量体系是容易让人有些莫名其妙的。

然而，基于 ISTQB 基本测试过程的观点来看，可能这样做更容易一些。比如说，如果我们是在测试一个安全关键的受美国联邦航空局 DO-178B 标准（我马上就要讲到该标准）限制的航空电子系统软件，那么我们的入口准则可能对测试覆盖率提出了一些特殊的要求。所以，在测试设计期间，我们会使用到测试设计的相关技术。在测试实现期间，将利用测试测量技术保证足够的覆盖率。

除了以上讲到的两个主要部分之外，该标准还提到了两个附录。附录 B 通过一些生动的例子来说明如何将前述两个小节中的枯燥内容应用到现实情况中。附录 A 涉及到了关于过程的一些考虑，可能最接近于我们比较关心的内容，其中讨论了如何按照该标准中给出的测试过程将该标准应用到实际的项目中。将这些过程对应于 ISTQB 的基本测试过程，可以得出以下结论：

- 在 ISTQB 中测试分析和设计连同测试实现部分等同于 BS 7925/2 中的测试规格。
- BS 7925/2 的测试执行，逻辑上对应 ISTQB 中的测试执行部分。但是需要注意的是，在 ISTQB 中只将其视为一个大的活动——测试实现和执行中的一部分。同时 ISTQB 把测试记录当作测试执行的一部分，而 BS 7925/2 将测试记录视为一个独立的过程。
- BS 7925/2 在其定义的测试过程中将检验测试的完成作为测试活动的最后一个步骤，大致相当于 ISTQB 中的评估出口准则和报告。

美国联邦航空局提出了一个关于航空电子系统的 DO-178B 标准，在欧洲叫做 ED-12B 标准。该标准根据一个失效的潜在影响提出了一个重要性级别的概念。基于不同的重要性级别，需要该级别达到指定的白盒测试覆盖率，如表 2-1 所示。

表 2-1　FAA DO 178B 强制要求的覆盖率

临　界	失效的潜在影响	测试必须覆盖的范围
A 级：灾难性的	软件的失效会导致系统灾难性的故障	修正条件/判定、判定和语句
B 级：危险的/严重的	软件的失效会导致系统危险的、严重的故障	判定和语句
C 级：主要的	软件的失效会导致系统的主要部件出现故障	语句
D 级：次要的	软件的失效会导致系统的次要部件出现故障	无
E 级：没有影响的	软件的失效不会导致任何系统故障	无

pin51.com 最新最全软件测试资料 拼吾爱

下面对表 2-1 的内容稍做解释：

重要性级别 A，或者说灾难性的级别，适用于由于软件的失效导致系统出现灾难性故障的情况，对于这样的重要性，该标准要求白盒测试要覆盖到修正条件/判定、判定和语句。

重要性级别 B，或者说危险的/严重的级别，适用于由于软件的失效导致系统出现危险的、严重的故障的情况，该标准要求白盒测试要覆盖到判定和语句。

重要性级别 C，或者说主要的级别，适用于由于软件的失效导致系统的主要部件出现故障的情况，该标准要求白盒测试要只覆盖到语句即可。

重要性级别 D，或者说次要的级别，适用于由于软件的失效导致系统的次要部件出现故障的情况，该标准对白盒测试的覆盖率不做要求。

最后，重要性级别 E，或者说没有影响的级别，适用于由于软件的失效不会导致任何系统故障的情况，该标准对白盒测试的覆盖率不做要求。

这是很有意义的一组定义。您将会更多地关注于对那些影响飞行安全的软件，比如方向舵和副翼控制组件，而不是把大部分精力放在那些与飞行安全无关的软件，比如说视频娱乐系统。然而，通过一维的白盒测试度量法的测量来决定我们对系统的信心程度存在一定风险。覆盖率的度量也是对信心程度的一种测量，确实没错，但是我们还需要使用多维的覆盖度量，无论是白盒还是黑盒。

顺便提一句，如果您对这部分内容感到有些困惑，其实这里讨论的所有关于白盒测试覆盖率的度量已经在基础级大纲的第 4 章中讲过了。如果您记不起来的话，那么就请温习一下基础级大纲及相关书籍中关于白盒测试覆盖率度量的相关章节。

2.5.9 度量元

最后，在 ISTQB 的基本测试过程中，在测试实现和执行阶段可以使用什么度量元和度量呢？当然，度量元的选取是因人而异的。

在测试实现期间，典型的度量元是测试环境已配置的百分比、测试数据记录加载的百分比、测试用例自动化的百分比。

在测试执行期间，典型的度量元是测试条件被覆盖的百分比、测试用例被执行的百分比等。

我们可以利用任务的分解结构来追踪和度量测试实现和测试执行的任务，这对于确认我们是否是按照预算和进程安排进行测试很有帮助。

请注意我们这里所讨论的度量元是用来测量某些过程（即工作的进展）的完整性。而报告测试的过程中，则需要使用一组不同的度量元。

2.6 评估出口准则和报告

学习目标：

（K3）根据一组给定的度量结果，确定是否符合测试完成准则。

从某种意义上讲，对于出口准则的评估和报告测试结果是一种测试管理活动。在关于高级测试管理的书中，我们将对如何进行测试结果分析、制图、陈述和报告的方法进行考察。这些活动将被视为测试进度监控活动的一部分。

然而，从另外一种意义上讲，在这个过程中有一个重要的部分是测试分析师的职责。作为一个测试分析师，本书我们最关心两个重要部分。第一，我们需要收集必要的信息支撑测试管理层报告测试结果。第二，我们需要测量完成的进度，相当于我们需要检测实际测试与计划的偏离（当然，如果我们确实检测到与计划的偏离，应该由测试经理开展能使我们回归正轨的测试控制活动）。

基于出口准则测量测试的完整性，并生成测试报告的相关信息，可以测量如下测试执行过程的属性：

- 测试条件、测试用例或是计划、执行、通过或未通过的测试规程的数量；
- 按照严重程度、优先级、状态或是其他因素分类的缺陷的总数；
- 被提议的、被接受的和被测试的变更要求（数量）；
- 计划的及其相应的实际的成本、进度和工作量（effort）；
- 质量风险，包括已得到消减的和遗留待解决的；
- 由于事件的阻碍（blocking）而浪费的时间；
- 确认测试和回归测试结果。

可以利用任务的分解结构来追踪评估和记录里程碑事件，这对于确定是否是按照预算和进度安排进行测试很有帮助。

本章后面部分，我们将看一组关于测试度量和测度的例子，可以评定（根据出口准则）项目处于什么状态。大部分例子是从 RBCS 帮助某客户进行测试的实际案例研究中得出的。

2.6.1 测试套件总结

图 2-3 是一个测试套件的总结工作表。该工作表总结了逐个测试的、在前述小节中已描述过的测试日志信息。作为一个测试分析师，可以利用这个工作表，追踪到目前为止测试执行中的一些重要属性：

- 测试用例的进展；
- 测试用例的通过/未通过率；
- 测试套件缺陷的优先级/严重程度（加权失效数量）；
- 挣值（已获取价值）。

就这点而论，这张表对测试分析师掌握整体测试工作的状态是很有帮助的。

在图 2-3 的最左侧，可以看到两栏，测试套件的名称和其包含的测试用例的数量。再次说明，上述测试用例的具体细节都应该可以在某个测试日志的某个部分找到。

在中间靠左的位置，可以看到通用标题为“完成计划测试的”的 4 列，它们是通过这轮测试，已完成测试的数量。

对于每个测试套件的失效加权数量，可以在本表的中部位置找到，它提供了一个关于测试套件检测到的缺陷数量的度量。每个缺陷的加权由其优先级和严重程度来决定，

pin51.com 最新最全源码分享下载 拼吾爱

测试套件摘要
第二轮 test pass

	完成计划测试的						测试计划未完成的				挣值			
测试套件	测试用例总数	数量	跳过	通过	未通过	失效的加权值	数量	排队等待的	IP	被阻隔的	计划花费的小时数	实际花费的小时数	成本花费比率(%)	用例执行比率(%)
功能性	14	14	0	4	10	10.6	0	0	0	0	36.0	49.0	163%	100%
性能	5	4	0	1	3	8.7	1	1	0	0	7.0	11.5	164%	80%
可靠性	2	2	2	0	0	0.0	0	0	0	0	0.0	0.0	0%	0%
健壮性	3	1	0	0	1	0.5	2	2	0	0	12.5	8.0	64%	33%
可安装性	4	0	0	0	0	0.0	4	4	0	0	72.0	0.0	0%	0%
本地化	8	1	0	0	1	0.5	7	0	7	0	128.0	22.0	17%	13%
安全性	4	2	0	2	0	1.1	2	2	0	0	17.0	8.5	50%	50%
文档	3	1	0	0	1	4.0	2	2	0	0	28.0	15.0	54%	33%
集成	4	4	0	3	1	1.0	0	0	0	0	8.0	12.5	156%	100%
易用性	2	0	0	0	0	0.0	2	0	2	0	16.0	0.0	0%	0%
探索性	6	0	0	0		0.0	6	6	0	0	12.0	0.0	0%	0%
总数	55	29	2	10	17	26.4	26	17	9	0	336.5	126.5	38%	51%
百分比	100%	53%	4%	18%	31%	N/A	47%	31%	16%	0%				

图 2-3　测试套件摘要表

只有严重程度为 1，优先级为 1 的缺陷能得到一个全（1 或 100%）加权，严重度和优先级较低的情况甚至缺陷的加权可被降低至 0.04（或 4%）。所以这是一个关于技术风险，即找到问题可能性的度量，常与每个测试套件的历史发现（检测到缺陷的）数据相关。

在中间靠右的位置，您会看到通用标题为“测试计划未完成”的 4 列，它们是通过这轮测试，未完成测试的数量。顺便解释一下，IP 指“进行中的（in processing）”。

最后，在该表的右侧，会看到总标题为“挣值”的 4 列。挣值是项目管理中一个简单的概念。它是指在一个项目中，我们消耗资源完成任务，如果任务完成的百分比近乎与资源使用的百分比相等，说明我们的测试在正常进行；如果任务完成的百分比大于资源使用的百分比，说明测试尚在预算之内；如果任务完成的百分比小于资源使用的百分比，说明测试已经超出预算了。

类似的，从进度安排来讲，任务完成的百分比与项目已消耗的时间的百分比应该近乎相等，或者从工作量的角度讲，如果任务完成的比率超过了进度安排的百分比，我们应该很高兴；而如果任务完成的比率超过了进度安排的百分比，我们则该担忧了。

在测试执行过程中，我们可以把测试用例或是测试规程当作我们的基本任务。用于手动测试的资源通常计量为进行测试需要花费的人工数，如人/小时数。该表格即是通过这样的计量方法来表述每个测试套件的挣值。

花点时间来研究一下图 2-3，想一想您会如何将其利用在现在或是将来的项目中。

2.6.2　缺陷的分解

分析缺陷（或是漏洞）报告有很多种方法。作为一名测试分析师、测试经理和测试

顾问，可亲身体会到分析是永无止境的。作为一名专业测试员，就应像医生正确而恰当地分析血液样本一样，正确而恰当地分析缺陷报告。

众多方法的其中之一是，根据缺陷的严重度、优先级或是二者的组合分解的数量来检验缺陷报告。如果您是使用数值刻度来表示严重度和优先级的，那么易行的方法就是，把这两种度量相乘，得到一个缺陷总体重要程度的权重度量，正如刚才在 2.6.1 节的图 2-3 中看到的。

图 2-4 中的图表正是一个应用该方法的例子，能用它做什么呢？首先，可以将这个表与之前的项目进行比较以了解我们是处于一个更好或是更差的状况。请记住，该分布会在项目生命周期期间一直变化。理想情况是，这个分布图（峰值，或曲线高的部分）应该逐步向高优先级的缺陷方向偏移，至少是在我们做正确的基于风险的测试时，第 3 章将进一步探讨这个问题。

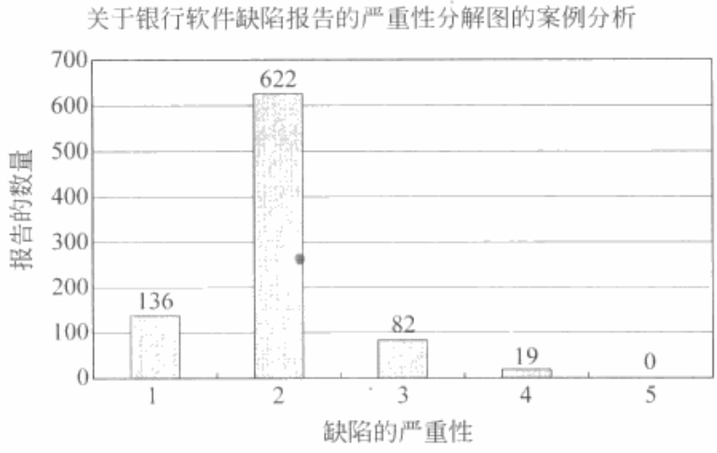


图 2-4 按照严重程度划分的缺陷的分类

图 2-4 表明，严重度为 1 和 2 的缺陷很多。通常情况下，严重度为 1 指数据缺失，严重度为 2 指因为相关功能缺失，同时没有替换方案。不管怎样，对我们而言都是坏消息。

这个图表告诉我们，作为一个专职的测试员，应该从这些严重度为 1 或是 2 的测试报告中随机抽取 10 或 20 个样本，看看严重度是否在膨胀（夸大其词），我们对严重度的分类是否准确？如果答案是“不”的话，那么给测试经理的报告是存在偏差甚至危言耸听的，这将给测试经理带来麻烦。

正如之前提到过的，如果我在进行基于风险的测试，这个图表的内容很有可能是要在项目的前 1/4 时间段需要被关注。我经常告诉所有的测试员，先找到具有威胁性的东西。而测试团队此时要做的，就是给出正确的严重度分类。

2.6.3 确认测试未通过率

作为测试分析师，我们只能借助找到的缺陷进行分析。但愿大部分检测出的缺陷，被修正后会再次返回给我们。此时我们需要进行确认测试，验证这些缺陷是否真的被修正了。有多少缺陷未通过确认测试？人们通常会觉得这个数量不大，但实际上呢？图 2-5

pin51.com 最新最全源代码 拼吾爱

就用图形的方式，给出了在一个实际项目中关于这个问题的答案。

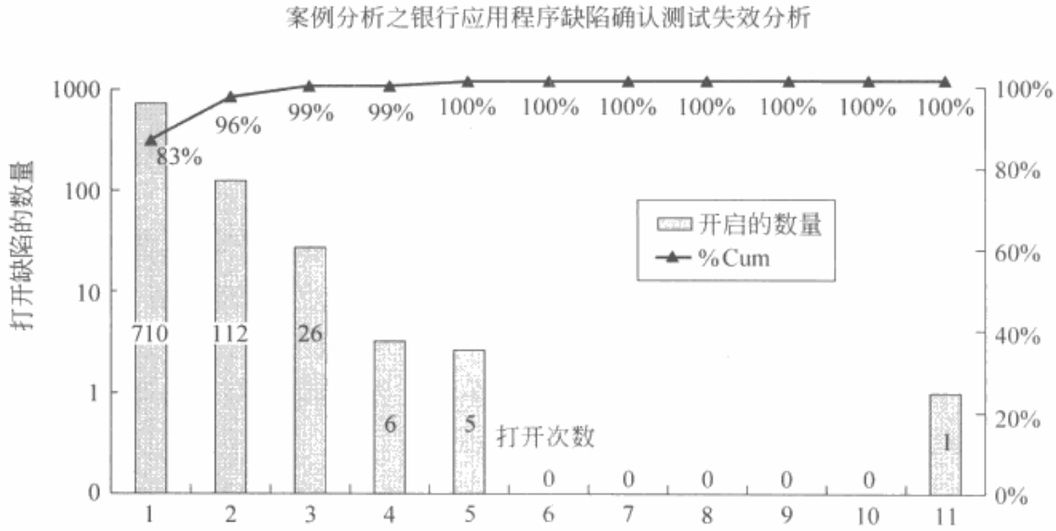


图 2-5　确认测试失效分析

在这个银行项目中，可以发现有相当多已经被修正的缺陷在确认测试中还是未通过。我们每次在 6 个缺陷报告中至少要完全重开（reopen）其中的一个。这浪费了很多的时间，很可能会延迟计划的进度安排，研究一下图 2-5，想想在测试执行阶段，您会如何利用这些信息。

2.6.4　系统测试出口评审

最后，再来看一个案例分析。图 2-6 中是 RBCS 曾经提供服务的一个互联网应用程序的项目中出口准则的摘要。您会发现已把对出口准则的分级当作系统测试出口评审的一部分。

系统测试出口评审

根据测试计划，系统测试将在以下准则均满足的情况下结束：

1. 系统测试入口会议中承诺的所有的设计、实现和特性，代码，单元测试，或者按照拟定的系统测试退出日期完成，或者分别在拟定日期后的四周、三周、三周内完成。

STATUS：RED。音频和演示功能已经在三周前进入系统测试阶段。调制解调器功能也已经在三周前进入系统测试阶段。在（软硬件）冲突的临界状态，摘机检测存在显著变更。

2. 在过去的三周内，服务器的软件和硬件都没有出现紧急状况、崩溃、停机、卡机的现象或是未预料的进程终止，以及其他形式的进程停止。

STATUS：YELLOW。服务器没有死机，但是我们还没有完成之前计划的翻转测试（Tip-over test）和失效备援测试（fail-over test），所以无法确信服务器在负荷峰值或是极限的条件下是否稳定。

3. 在过去至少三周中，生产设备已经被用于所有的系统测试执行。

STATUS：GREEN。除了如上提到的调制解调器的问题，硬件已经很稳定了。

图 2-6　系统测试出口评审的案例分析

图中的 3 个出口准则中，在这里每个都按照以下三阶段状态分类。

- 绿色：完全满足，几乎没有风险存在。

pin51.com 拼客发 资源分享

- 黄色：没有完全满足，有风险存在，但是在可以接受的范围内。
- 红色：根本没有满足，会引发实质的风险。

当然，对于黄色和红色类别，您肯定希望获得更多的信息和数据。

2.6.5 标准

最后，再来看一下 IEEE 829 标准中适用于测试过程相关的模板。IEEE 829 测试文档标准包含一个关于测试总结报告的模板。

测试总结报告描述的是关于某个具体测试级别或测试阶段的结果。在 IEEE 829 的模板中包含以下几个部分：

- 测试总结报告标识；
- 总结（例如，测试了什么，结论是什么等）；
- （测试计划、用例、规程的）变化；
- 全面评估；
- 结果总结（例如，最后的度量，计数）；
- （每个测试项通过/未通过准则）的评估；
- 活动的总结（资源的使用、效率等）；
- 审批。

测试总结报告可以在测试执行期间交付，以作为项目现状报告或会议中的一部分。还可以作为测试结束活动的一部分，在一个测试级别结束时使用。

2.7 评估出口准则和报告练习

在互联网应用程序的项目中，所有在实际项目中使用到的准入原则会在下面的子章节中给出。一直到该项目结束的时候，测试团队都按照以下状态级别对各准则进行分类：

- 绿色：完全满足，几乎没有风险存在。
- 黄色：没有完全满足，有风险存在，但是在可以接受的范围内。
- 红色：根本没有满足，会引发实质的风险。

> **ISTQB 术语**
>
> **测试结束**：从已完成的测试活动中收集数据，总结基于测试件及相关事实和数据的测试结束阶段，包括对测试件的最终处理和归档，以及测试过程评估（包含测试评估报告的准备）。
>
> **测试总结报告**：总结测试活动和结果的文档。也包括对测试项是否符合出口准则进行的评估。

您可以在每条准则的 STATUS 部分看到我们给出的该准则的评级。

在系统测试退出评审会议上，我们把对这些准则的评估作为议程的一部分。我曾经主持过这样的会议，并对每条准则为测试团队进行了走查（或解释）。您可以想象得到，

pin51.com 最新最全 拼吾爱

在状态被评估为红色的准则上，需要解释的远比状态被评估为黄色或绿色的多。

然而，每个评估都需要给出详细的解释，甚至需要更多的信息和数据支持。因此，对于每条准则，您想收集什么类型的数据和其他信息来支持每条准则评估的结果，就应该在每条准则的状态评估中同时给出。

2.7.1 系统测试退出评审

根据测试计划，系统测试在如下标准满足时才能宣告结束：

1. 系统测试入口会议中承诺的所有的（1）设计、实现和特性；（2）代码；（3）单元测试，或者按照拟定的系统测试退出日期完成，或者分别在拟定日期后的四周、三周、三周内完成。

STATUS：RED。音频和演示功能已经在三周前进入系统测试阶段。调制解调器功能也已经在三周前进入系统测试阶段。在软硬件冲突临界状态，摘机检测存在显著变更。

2. 在过去的三周内，服务器的软件和硬件都没有出现紧急状况、崩溃、停机、卡机的现象或是未预料的进程终止，以及其他形式的进程停止。

STATUS：YELLOW。服务器没有死机，但是我们还没有完成之前计划的翻转测试和失效备援测试，所以我们无法确信服务器在负荷峰值或是极限的条件下是否稳定。

3. 在过去至少三周中，生产设备已经被用于所有的系统测试执行。

STATUS：GREEN。除了如上提到的调制解调器的问题，硬件已经很稳定了。

4. 在过去至少三周内，没有客户系统因为一个更新失败而无法操作。

STATUS：YELLOW。在更新的过程中，没有一个系统会变得永久性地不可操作，但我们的确观察到在更新的过程出现过系统崩溃的现象，并需要系统重启才能清除该错误。

5. 服务器进程已经在没有安装漏洞补丁、人工干预或是调整配置文件的情况下运行两周。

STATUS：RED。在过去的两周内，经变更委员会核准的服务器配置已经变动了多次。

6. 测试团队已经对所有候选发布的硬件和已发布的设备、服务器端、客户端的软件执行了原计划的测试。

STATUS：RED。计划测试采购和订单履行系统，但是因为系统尚未完成，导致无法进行相关工作。我们刚刚收到了候选发布的（软件系统）构建版本，要完成相关测试还需要两周。此外，经变更委员会核准，服务器每过几天就会发生变更，并且服务器群中又新增了一个负载平衡器。这些关于服务器的变更，使得过去一周半的时间内我们无法进行大数据量、翻转测试和失效备援测试。最后，我们根本没有机会测试服务器的安装和启动进程，因为我们从未收到任何关于如何执行这些任务的文档。

7. 测试团队已经对测试生命周期中产生的缺陷报告中所有优先级为1和2的缺陷，在候选发布的硬件和已发布的设备、服务器端、客户端的软件测试上重新测试一次。

STATUS：RED。对所有候选发布的硬件和软件的测试已经被限制到一周内完成，没有时间重测所有的缺陷了。

8. 开发团队已经对所有“必须修改”的缺陷做出修改，哪些缺陷“必须修改”由项

目管理团队决定。

STATUS：RED。根据所附的开启/关闭（open/close）图表和“自 11 月 9 日发现的缺陷”报告，显示我们在这个产品中持续发现新的缺陷。有个好消息就是，优先级为 1 的缺陷所占比例已呈下降趋势。根据关闭周期的图表显示，平均需要两周（优先级为 1 的缺陷需要三周）才能关闭一个问题报告。此外，根据开启/关闭的统计图表显示，在累计开启数和累计关闭数中间存在很大的质量漏洞。这让人很难相信该产品中没有存在着一定数量的缺陷，导致产品整体缺乏外观及质量的可靠性。最后，请注意网络指南和电子商务的问题是设计的问题——所选定的浏览器与很多网络都不兼容——使得这些问题更加令人担忧。

9. 测试团队已经确认过在缺陷跟踪系统中所有的问题都是关闭或是延缓（deferred）状态，适当情况下，一些问题已经通过确认测试和回归测试验证。

STATUS：RED。一个重大的质量漏洞已经存在数月了。由于针对候选发布构建的测试时间有限，回归风险显著。

10. 开启/关闭曲线表明产品已达到了稳定性和可靠性的目标。

STATUS：RED。优先级为 1 的被发现缺陷曲线已经趋于稳定，但被发现缺陷总数的曲线并非如此。此外，很多错误导致需要重启系统表明系统仍然会每隔 8 小时崩溃一次，其稳定性不比装有 Windows 95/Windows 98 的笔记本好多少（曾有广告称系统的稳定性比 PC 要好）。

11. 项目管理团队同意系统测试的最后一个周期中测试的产品，将可以满足客户合理的质量预期。

STATUS：YELLOW。我们目前尚未运行足够的测试套件，所以无法对产品的整体质量给出一个好的评估。

12. 项目管理团队召开系统测试结束会议，同意我们已经完成系统测试。

STATUS：在进行中。

2.8 评估出口准则和报告练习解答

在如下所列的所有标准后面又增加了一个叫做“附带数据和信息”的部分。在本章节，您会看到针对这个练习的我个人的解决方案。这些方案是基于在开会时我已经掌握的附带数据和信息以及如果我当时知道的像我现在知道的这么多时，我希望附带的数据和信息。

1. 系统测试入口会议中承诺的所有的设计、实现和特性，代码，单元测试，或者按照拟定的系统测试退出日期完成，或者分别在拟定日期后的四周、三周、三周内完成。

STATUS：RED。音频和演示功能已经在三周前进入系统测试阶段。调制解调器功能也已经在三周前进入系统测试阶段。在软硬件冲突临界状态，摘机检测存在显著变更。

ADDITIONAL DATA AND INFORMATION：在系统测试的启动会议上做出的具体承诺。音频功能、样片功能、调制解调器和摘机检测功能的交付日期。

2. 在过去的三周内，服务器的软件和硬件都没有出现紧急状况、崩溃、停机、卡机

pin51.com 最新最全 拼吾爱

的现象或是未预料到的进程的终止，以及其他形式的进程的停止。

STATUS：YELLOW。服务器没有死机，但是我们还没有完成之前计划的翻转测试和失效备援测试，所以我们无法确信服务器在负荷峰值或是极限的条件下是否稳定。

ADDITIONAL DATA AND INFORMATION：翻转测试和失效备援测试完成百分比的度量元。由于未运行这两个测试会遗留哪些特定的质量风险的具体细节。

3. 在过去至少三周中，生产出来的设备已经完成所有的系统测试。

STATUS：GREEN。除了如上提到的调制解调器的问题，硬件已经很稳定了。

ADDITIONAL DATA AND INFORMATION：无。好消息无须解释。

4. 在过去至少三周内，没有客户系统因为更新的失败而无法操作。

STATUS：YELLOW。在更新的过程中，没有一个系统会变得永久性地不可操作，但我们的确观察到在更新的过程出现过系统崩溃的现象，并需要系统重启才能清除该错误。

ADDITIONAL DATA AND INFORMATION：具体的细节，在缺陷报告中就系统崩溃有哪些描述。

5. 服务器进程可以在没有安装漏洞补丁，人工干预或是调整配置文件的情况下运行两周。

STATUS：RED。在过去的两周内，经变更委员会核准服务器的配置已经变动了多次。

ADDITIONAL DATA AND INFORMATION：在最后的变更前都运行了哪些测试的清单，以及对每个变更给测试带来的风险评估。（备注：除非您有很好的可追溯性信息，否则制作这个测试清单和风险评估的工作量会非常大。）

6. 测试团队已经对所有候选发布的硬件和已发布的设备、服务器端、客户端的软件执行了原计划的测试。

STATUS：RED。我们计划测试采购和订单履行系统，但是因为系统尚未完成，导致我们无法进行相关工作。我们刚刚收到了候选发布的（软件系统）构建版本，要完成相关测试还需要两周。此外，经变更委员会核准，服务器每过几天就会发生变更，并且服务器群中又新增了一个负载平衡器。这些关于服务器的变更，使得过去一周半的时间内我们无法进行大数据量、翻转测试和失效备援测试。最后，我们根本没有机会测试服务器的安装和启动进程，因为我们从未收到任何关于如何执行这些任务的文档。

ADDITIONAL DATA AND INFORMATION：采购和订单履行系统中跳过的测试清单，以及和这些测试相关的风险。由于最后一轮测试通过的时间紧迫而计划跳过的测试清单，以及和这些测试相关的风险。自最后一次大数据量、翻转测试和失效备援测试以来服务器的变更清单，以及由变更引发的可靠性风险的评估。（重申一下，这可能需要庞大的工作量。）对服务器安装和启动进程功能跳过的测试清单，以及由此引发的风险。

7. 测试团队已经对所有候选发布的硬件和设备、服务器端、客户端的软件测试时产生的缺陷报告中，优先级为 1 和优先级 2 的缺陷重新进行一遍测试。

STATUS：RED。对所有候选发布的硬件和软件的测试已经被限制到一周内完成，没有时间重测所有的缺陷了。

ADDITIONAL DATA AND INFORMATION：项目中所有优先级为 1 和优先级 2 的

缺陷报告的清单，以及对这些缺陷由于在变更引发的回归测试中没有被测出来而重新进入系统的风险评估（再次重申，这将是很庞大的工作量）。

8. 开发团队已经对所有“必须修改”的缺陷做出修改，哪些缺陷“必须修改”由项目管理团队决定。

STATUS：RED。根据所附的开启/关闭图表和“自 11 月 9 日发现的缺陷”报告，显示我们在这个产品中持续发现新的缺陷。有个好消息就是，优先级为 1 的缺陷所占比例已呈下降趋势。根据关闭周期的图表显示，平均需要两周（优先级为 1 的缺陷需要三周）才能关闭一个问题报告。此外，根据开启/关闭的统计图表显示，在累计开启数和累计关闭数中间存在很大的质量漏洞。这让人很难相信该产品中没有存在着一定数量的缺陷，导致产品整体缺乏外观及质量的可靠性。最后，请注意网络指南和电子商务的问题是设计的问题——所选定的浏览器与很多网络都不兼容——使得这些问题更加令人担忧。

ADDITIONAL DATA AND INFORMATION：开启/关闭图表，从 11 月 9 日以来所有缺陷的清单、结束时期的图表和一些重要的与浏览器不兼容的网址的清单（注意，这两张图表出现在本书配套的高级测试经理培训课程中）。

9. 测试团队已经确认过在缺陷跟踪系统中所有的问题都是关闭或是延缓状态，适当情况下，一些问题已经通过确认测试和回归测试验证。

STATUS：RED。一个重大的质量漏洞已经存在数月了。由于针对候选发布构建的测试时间有限，回归风险显著。

ADDITIONAL DATA AND INFORMATION：非关闭或是延缓状态的缺陷报告的清单，按照优先级排列。计划将不会对候选发布版本进行的测试的清单，以及由此计划引发的风险和不运行每个测试可能引发的风险。

10. 开启/关闭曲线表明产品已达到了稳定性和可靠性的目标。

STATUS：RED。优先级为 1 的被发现缺陷曲线已经趋于稳定，但被发现缺陷总数的曲线并非如此。此外，很多错误导致需要重启系统表明系统仍然会每隔 8 小时崩溃一次，其稳定性不比装有 Windows 95/Windows 98 的笔记本好多少（曾有广告声称系统的稳定性比 PC 要好）。

ADDITIONAL DATA AND INFORMATION：开启/关闭图表，包括一份针对优先级仅为 1 的缺陷的和一份针对所有缺陷的。需要重启的错误的运行表，也就是说显示每天有多少系统崩溃需要重启的趋势图。

11. 项目管理团队同意系统测试的最后一个周期中测试的产品，将可以满足客户合理的质量预期。

STATUS：YELLOW。我们目前尚未运行足够的测试套件，所以无法对产品的整体质量给出一个好的评估。

ADDITIONAL DATA AND INFORMATION：尚未对候选发布版本进行的所有测试的清单，以及由此引发的风险（应该细化到每个测试）。

12. 项目管理团队召开系统测试结束会议，同意我们已经完成系统测试。

STATUS：在进行中。

ADDITIONAL DATA AND INFORMATION：无。

pin51.com 拼吾爱 最新最全源分享下载

2.9 测试结束活动

学习目标:

能回想起该部分内容即可。

本章节的概念主要适用于测试经理。在这个章节对测试分析师在学习目标上不做要求。在认证考试的学习课程中，在高级大纲中的第 2 章可找到相关内容，对这部分内容仅要求熟悉并能回想起即可。

2.10 认证考试模拟题

在每个章节的最后，您可以试着做几道认证考试模拟题以加强对材料的记忆和理解，准备 ISTQB 高阶测试分析师的考试。

1. 辨别如下所列的项目中，哪些可以作为一个测试用例第一次运行的测试准则？
 A 事件报告　　B 需求规格说明
 C 测试总结报告　　D 既有系统
2. 假设您是一个测试分析师，正在负责一个银行项目，将现有的自动提款机升级为允许用户用支持的信用卡提现。在测试设计阶段，您发现了一个需求规格说明与设计规格说明中的关于支持信用卡的清单存在差异。这是以下哪种测试活动的实例？
 A 将测试设计作为一种静态测试技术
 B 存在于需求规格书中的一个缺陷
 C 存在于设计规格书中的一个缺陷
 D 在项目中测试设计开始得过早
3. 以下哪些不总是测试执行阶段的预置条件？
 A 正确配置的测试环境　　B 完整规定的测试规程
 C 管理已被认可的缺陷的过程　　D 测试准则
4. 假设您是一个测试分析师，正在负责一个银行项目，将现有的自动提款机升级为允许用户用该银行支持的信用卡预支现金。在测试计划中的一个出口准则中要求从所有支持的信用卡中成功提现至少 500 欧元，并将这一过程进行记录。正确的支持的信用卡清单为，American Express（美国运通卡）、Visa（威士卡）、Japan Credit Card（日本信用卡）、Euro-card（欧元银行卡）和 MasterCard（万事通）。
 测试执行后，提现结果如下：
 - American Express 允许提现上限为 1000 欧元。
 - Visa 允许提现上限为 500 欧元。
 - Eurocard 允许提现上限为 1000 欧元。
 - MasterCard 允许提现上限为 500 欧元。

下面哪种说法是正确的？

A 不符合出口准则，因为 American Express 和 Eurocard 的提现上限超出了标准中指定的

B 不符合出口准则，一方面因为 American Express 和 Eurocard 之间的差异，另一方面因为 Visa 和 MasterCard 之间的差异

C 符合出口准则，因为所有支持的银行卡都可以满足最低提现限额

D 不符合出口准则，因为没有记录 Japan Credit Card 的测试结果进行记录

pin51.com 拼吾爱 最新最全源码素材分享

第3章

测试管理

> “如果林中的一棵树倒了，而且已经把它卖掉了，那么它有质量可言吗？”
>
> 连环漫画中 Dilbert 秃头老板，利用篡改自著名的禅学心印的一句话，答 Ratbert 在测试工程师声明中关于测试结果不能保障产品发行的言论。

高级大纲中的第 3 章与测试管理有关。该章讨论了在测试过程自始至终所有的测试管理活动，并介绍了关于测试风险的一些需要考虑的事情。一共包括 11 个小节：

1. 概述。
2. 测试管理文档。
3. 测试计划文档模板。
4. 测试估算。
5. 测试进度计划。
6. 测试进度监视与控制。
7. 测试的商业价值。
8. 分布式测试、外包测试和内包测试。
9. 基于风险的测试。
10. 失效模式和影响分析（FMEA）。
11. 测试管理的相关问题。

让我们通过对每个章节的深入讨论来看看它们与测试分析的关系。

ISTQB 术语

测试管理：计划、估计、监控和控制测试活动，通常由测试经理来执行。

3.1 概　　述

学习目标：

能回想起该部分内容即可。

本节会主要集中于测试管理的相关话题。因此，它主要是高级大纲测试经理认证考试的内容。本书主要是针对测试分析师的，所以本章中所涉及的内容对测试分析师的要求比较简单，大部分只要求能回想起相关内容即可。

然而，作为测试分析师，有一个重要的部分，您需要对其有很好的理解，就是基于风险的测试。本章将会学习到，如何进行风险分析，然后按照风险给测试进行排序。这是测试分析师在进行基于风险的测试时非常关键的任务。

在高级大纲中，本章还提到了可以为测试经理所使用的测试文档模板，主要来自 IEEE 829 标准。作为一名测试分析师，您需要对这一标准非常了解。如果您计划参加 ISTQB 高级测试分析师认证考试，请记住关于如何使用 IEEE 829 的模板和测试管理等所有资料都可以在基础级大纲的第 5 章中找到。

3.2　测试管理文档

学习目标：

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。

ISTQB 术语

测试方针：描述有关组织测试的原则、方法和主要目标的高级文档。

测试策略：一个高级文档，该文档定义了需要对程序（一个或多个项目）执行的测试级别和需要进行的测试。

测试级别：统一组织和管理的一组测试活动。测试级别与项目的职责相关联。例如，测试级别有组件测试、集成测试、系统测试和验收测试。

级别测试计划：通常用于一个测试级别的测试计划。

主测试计划：通常针对多个测试级别的测试计划。

测试计划：描述预期测试活动的范围、方法、资源和进度的文档。它标识了测试项、需测试的特性、测试任务、任务负责人、测试人员的独立程度、测试环境、测试设计技术、测试的进入和退出准则和选择的合理性、需要紧急预案的风险，是测试策划过程的一份记录。

不过，基础级大纲中涉及到了关于测试管理的内容，包括关于测试管理文档的话题。所有在基础级大纲中涉及到的内容都在认证考试范围。此外，高级大纲中所有要求能够回想起这部分内容的章节也都在考试的范围。

所以，如果您正在为准备认证考试而学习的话，那么，您需要参照高级大纲中的第 3 章中回想和熟悉相关内容，并复习基础级大纲中关于测试管理文档的内容。

pin51.com 最新最全源码分享下载 拼吾爱

3.3 测试计划文档模板

学习目标:

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。

在第2章，回顾了IEEE 829测试文档标准的部分内容。确切地说，我们研究了测试设计规格说明模板、测试用例规格说明模板、测试规程规格说明模板、测试总结报告模板和测试日志模板。不过，基础级大纲中涉及到了整个IEEE 829标准的内容，包括测试计划模板、测试项传递报告模板和事件报告模板。

所有在基础级大纲中涉及到的内容都在认证考试范围。此外，高级大纲中的所有要求能够回想起这部分内容的章节也在认证考试范围。所以，如果您正在为准备认证考试而学习的话，那么，您需要参照高级大纲中的第3章中回想和熟悉相关内容，并复习基础级大纲中关于测试管理文档的内容。

3.4 测 试 估 算

学习目标:

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。

不过，基础级大纲中的第5章作为测试管理的一部分涉及到了测试估算的内容。所有在基础级大纲中涉及到的内容都在认证考试范围。此外，高级大纲中的所有要求能够回想起这部分内容的章节也在认证考试范围。所以，如果您正在为准备认证考试而学习的话，那么，您需要参照高级大纲中的第3章中回想和熟悉相关内容，并复习基础级大纲中关于测试管理文档的内容。

3.5 测试进度计划

学习目标:

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。如果您正在为准备认证考试而学习的话，那么，您需要参照高级大纲中的第3章中回想和熟悉相关内容。

ISTQB术语

测试估算：对（例如，花费的工作量、完成时间、涉及的成本、测试用例的数目

等）这些可用的、即使可能不完整，不确定或嘈杂的输入数据近似结果的计算。

测试时间表：测试过程中活动、任务或事件的清单，说明活动、任务或事件开始和结束的日期和/或时间，和它们之间的相互依存关系。

测试点分析（TPA）：基于功能点分析的一种公式化测试估计方法。

宽带德尔菲法（Wideband Delphi）：一种专家测试评估的方法，旨在集团队成员的智慧来做精确地评估。

3.6 测试进度监视与控制

学习目标：

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。

不过，基础级大纲中的第 5 章作为测试管理的一部分涉及到了测试进度监控的内容。所有在基础级大纲中涉及到的内容都在认证考试范围。此外，高级大纲中的所有要求能够回想起这部分内容的章节也在认证考试范围。所以，如果您正在为准备认证考试而学习的话，那么，您需要在高级大纲中的第 3 章中回想和熟悉这部分内容，以及回顾在基础级大纲中关于测试进度监控的内容。

ISTQB 术语

测试监控：处理与定时检查测试项目状态等活动相关的测试管理工作。准备测试报告来比较实际结果和期望结果。

3.7 测试的商业价值

学习目标：

能回想起该部分内容即可。

本节内容主要适用于测试经理，对测试分析师不做要求。如果您正在为准备认证考试而学习，可以在高级大纲的第 3 章中回想和熟悉这部分内容。

3.8 分布式测试、外包测试和内包测试

学习目标：

能回想起该部分内容即可。

pin51.com 最新最全源码分享 拼吾爱

本节内容主要适用于测试经理，对测试分析师不做要求。如果您正在为准备认证考试而学习，那么，可以参照高级大纲中的第3章中回想和熟悉相关内容。

3.9 基于风险的测试

学习目标：

（K3）依据风险对测试用例选择、测试覆盖和测试数据进行优先级划分，并在测试进度和测试规程中进行相应的文档化。

（K2）列出领域测试中采用基于风险的方法进行计划和执行测试所需的全部活动。

风险是负面的或是预料之外的结果或事件的可能性。具体地讲，风险是任何可能会降低客户、用户、参与者或是利益相关者对产品质量或是项目成功信心的影响因素。

在测试的时候，我们主要关心两类风险。一类是产品或者质量风险。当一个潜在的问题主要影响产品的质量时，这个潜在的问题便称之为产品风险。我自己经常将产品风险称为质量风险。举一个质量风险的例子，一个潜在的可靠性方面的缺陷可能会导致系统在正常操作时崩溃。

另一类风险是项目或计划风险。当一个潜在的问题主要影响项目的成功时，这个潜在的问题便称之为项目风险。有些人常把项目风险叫做计划风险。举一个项目风险的例子，人力短缺的问题可能会导致项目延期。

风险在其重要性方面不尽相同。有很多种方法可以对风险的级别进行分类。最简单的方法是考虑以下两个因素：

- 问题发生的可能性；
- 问题发生后的影响。

问题发生的可能性，主要出于对技术的考虑，例如使用的编程语言、线路的带宽等。问题发生后的影响，主要出于对商业的考虑，例如因某一问题引发经济上的损失，某一问题影响到的用户或客户的数量等。

ISTQB 术语

风险：风险将会导致负面结果的因素。通常表达成可能的负面影响。

风险类型：用特定类型的测试能降低特定种类的风险。例如：可用性测试能降低因用户错误操作而引起的风险。

产品风险：与测试对象有直接关系的风险。

项目风险：与测试项目的管理和控制相关的风险，例如：缺乏配备人员、严格的期限、需求的变更等。

基于风险的测试：在项目初始阶段使用的一种测试方法，用来降低产品风险的级别并通知利益相关者产品风险的状态。这个方法包括了产品风险识别和产品风险在测试过程中的使用。

在基于风险的测试中，通常在识别风险项的同时，还会分析每个风险项相关的风险等级并用于指导我们的测试。事实上，如严格采用基于风险测试策略时，风险将是测试的主要依据。

风险可以从很多方面指导测试，以下所列出来的是最常见的 3 种：

- 第一种，在所有的测试活动中，测试经理和测试分析师按照风险的等级为各个质量风险项分配工作量。测试分析师根据风险的等级决定测试技术的严密程度和广泛性。测试经理和测试分析师执行测试活动的顺序与风险的排序刚好相反，把主要的注意力放在最重要的质量风险上，而再最后花一点时间在次要的风险上。最终，测试经理和测试分析师需要与项目团队一同努力确保缺陷的修正（所花费的精力）是与风险的级别相对应的。
- 第二种，在测试计划和测试控制阶段，测试经理要对所有已知的重要的项目风险给出减轻风险的方案和应急措施。项目风险等级越高，越要被周全地管理。
- 第三种，测试经理和测试分析师需要针对遗留的风险，记录测试的结果和项目的状态。例如，还有哪些测试未执行或是被略过了？执行了哪些测试？哪些测试通过了？哪些测试没有通过？哪些缺陷还未被修正或是重测？这些测试和缺陷是如何与风险回溯关联的？

iSTQB 术语

测试控制：当监测到与预期情况背离时，制定和应用一组修正动作以使测试项目保持正常进行的测试管理工作。

风险等级：风险的重要性，由风险的影响和可能性定义。风险级别能用于决定测试的强度。风险级别既能用定性的词（比如：高、中、低）表示，又能用定量的词表示。

当决定严格采用基于风险的测试策略时，要注意的是在一个项目中风险管理不是仅仅进行一次，刚才所提到的 3 种对风险的响应和其他可能的响应一样，应该（是个反复的过程）贯穿于生命周期的始终。特别是，应该通过运行测试并发现缺陷来降低质量风险，同时通过采取减轻风险的方案甚至应急措施来降低项目风险。在项目中，需要定期根据新获取的信息，重新评估风险和风险等级。这可能会使得我们重新为测试和缺陷划分优先次序，重新分配测试工作量，并采取其他测试控制措施。

为了帮助人们理解基于风险的测试，有时会将这种测试比作某种形式上的保险。在日常生活中，当你为某些潜在的风险担心时，你会买保险，当然你也肯定不会为你完全不担心的风险买保险。所以应该测试那些使我们忧虑的方面，发现缺陷，而不需要在那些无须忧虑的方面浪费时间。

不过，在这个比喻中，有一个潜在的误导，就是保险从业人员和精算师，可以将一些有充分依据的统计数据应用于定量的风险分析中。而通常，基于风险的测试主要依赖于定性分析，因为我们手上没有像保险公司掌握的那些数据。

在基于风险的测试中，您还需要注意许多其他引发风险的源头。某些系统会存在安

pin51.com 拼吾爱 最新最全资源分享

全风险，很多系统会存在商业和经济风险，很多系统会存在隐私和数据安全的风险，甚至还有技术、组织、政策等方面的风险。

3.9.1 风险管理

风险管理主要包括以下3种活动。

- 风险识别，找出项目中存在的项目风险和质量风险。
- 风险分析，通常根据风险发生的可能性和影响，评估每个已被识别的风险项的等级。
- 风险减轻，其实其更准确的叫法是“风险控制”，因为其由对各种风险采取的一系列减轻、应急、转移和接受的动作组成。

在某种意义上，以上这3种活动是按次序进行的，至少就它们何时开始而言，是这样的。它们分段进行，风险识别最先开始，然后是风险分析，当风险等级确定后，风险控制即开始。不过，由于项目中风险管理是持续的过程，所以事实上，风险识别、分析和控制都是循环进行的。

每个人对如何管理项目的风险见仁见智，包括什么是风险、风险的等级、哪些活动对风险的控制是恰当的。所以风险管理应该将所有的项目利益相关者包括在内。

在很多情况下，虽然不是所有的利益相关者会或愿意参与到风险管理中。在这种情况下，一些利益相关者会扮演其他利益相关者的代理人的角色。比如，在针对一个大众市场的软件开发过程中，市场团队可能会邀请一小部分潜在的顾客帮助辨别出哪些潜在的缺陷会严重影响到他们对软件的使用。在这种情况下，这一小部分的潜在顾客就是整个最终客户群的代理人。再比如，IT项目的业务分析师有时可以代表用户参与到某些风险分析环节，讨论哪些可能会出错，程度如何，而避免最终用户直接参与可能面临的窘境。

ISTQB术语

风险管理：对风险进行标识、分析、优先级划分和控制所应用的系统化过程和实践。

风险识别：使用技术手段（例如，头脑风暴（brainstorming）、检验表（checklists）和失败历史记录（failure history））标识风险的过程。

风险分析：评估识别出的风险以估计其影响和发生的可能性的过程。

风险减轻或是风险控制：为降低风险到或控制风险在指定级别而达成的决议和实施防范（度量）措施的过程。

由于测试分析师聚焦缺陷的思维模式，而具备很多特定的关于风险管理的专业知识。因此，无论何时他们都应该积极参与到风险管理中。事实上，在很多情况下，测试经理会主导质量风险分析，而测试分析师会在该过程中提供重要的技术支持。

在适当地概述了风险管理之后，下面进一步讨论这3种测试管理活动。

pin51.com 最新最全资源分享下载

3.9.2 风险识别

恰当的基于风险的测试，不仅需要识别产品的风险，还需要识别项目的风险。可以利用以下技术识别这两种风险：

- 专家咨询；
- 独立评估；
- 使用风险模板；
- 项目回顾；
- 风险专题讨论会和头脑风暴；
- 检查表；
- 借鉴过去的经验。

您可以使用一个独立的完整的过程识别项目和产品的风险。不过，应经常将识别这两种风险的过程分开进行，因为它们各自有不同的交付物。应把对项目风险的识别过程包含在测试计划过程中；同时，质量风险的识别也要在项目的早期进行。

也就是说，项目风险，不仅指测试而是指整个项目，通常被视为质量风险分析的副产品。另外，如果您使用需求规格说明、设计规格说明、用例或是其他的文档作为质量风险分析过程的输入，您应该期望在这些文档中找到缺陷作为另外一组副产品。它们都是很有价值的副产品，您应该计划如何捕获这些副产品，并提交给合适的人（以便这些缺陷得到适当后续处理）。

之前，我曾经鼓励你把所有可能的利益相关者的小组代表纳入到风险管理的过程中。对于风险识别活动，利益相关者参与的范围越多，风险的识别就越完整、精确和确切；而你在该过程中忽略的利益相关者的小组代表越多，你会漏掉的风险项越多，甚至会漏掉整个风险类别。

应该把这个过程进行到什么程度呢？这取决于使用的技术，通常应使用非常规的技术，风险识别在所有风险项都已明确时停止。而且，你必须尽可能详尽地描述所有风险项，从而为将来的风险分析和评估提供明确的可能性和影响程度。

更为正规的技术通常是向下游方向识别风险发生后的影响。这些影响包括对系统或者综合系统的影响，以及对潜在用户、顾客、利益相关者甚至是社会团体的影响。失效模式和影响分析，就是这种正式的风险管理技术的一个例子，通常用于安全关键系统和嵌入式系统。

另外一种正规的技术是向上游方向识别风险的来源。危险分析（Hazard Analysis）就是这种正式的管理技术的一个例子。我本人从来没有用过，不过我和一些利用该技术于医疗安全关键系统的客户谈起过相关话题。[1]

1 关于风险识别和分析技术更多的相关信息，可以参考我的另外两本书：Managing the Testing Process, 2e（既包含正规的技术也包括非正规的技术）或 Pragmatic Software Testing（关于一些非正式的技术）。另外，也可以在 Rick Craig 写的 Systematic Software Testing 和 Paul Gerrard 写的 Risk-based E-business Testing 这两本书中得到基于风险测试技术的其他观点。如果你使用失效模式和影响分析（FMEA）技术，那么我推荐你阅读 D.H. Stamatis 撰写的 Failure Mode and Effect Analysis。这本书在 Managing the Testing Process, 2e 中相关讨论基础上，对失效模式和影响分析技术进行了深入地探讨。

pin51.com 最新最全源码素材大全 拼吾爱

ISTQB 术语

失效模式和影响分析（FMEA）：一个系统的进行风险识别和标识可能的失效模式的系统方法，用来预防失效的发生。

失效模式、影响和危急程度分析（FMECA）：FMEA 的扩展，除了基本的失效模式和影响分析（FMEA），作为对 FMEA 的补充，还包括危急程度分析。危急程度分析技术用于分析失效模式发生的可能性及其后果的严重性并将其制成图表。对于具有高可能性和严重性的失效模式优先采取补救措施。

3.9.3 风险分析或风险评估

有关风险管理过程的下一个步骤在高级大纲中被称为风险分析。我比较喜欢把它叫做风险评估，因为对我而言，分析似乎不仅包含了风险的评估，还包括风险的识别。不管我们如何称呼这一行为，风险分析或是风险评估都会涉及到对已识别的风险的研究。我们通常希望将每个风险项恰当地分类，并为每个风险项指定一个适当的风险等级。

可以应用 ISO 9126 或是其他质量类别来划分风险项。风险项被归到哪个类别并不是特别重要，只要我们不会忽视这个风险项就好。不过，在复杂的项目或是对于大公司来讲，风险项的类别决定了将由谁来负责该风险项，此时恰当的分类变得尤为重要。

风险评估的另一项任务是决定风险的等级。这部分通常会涉及到风险发生的可能性和影响两个重要因子。一般来讲，发生的可能性是从技术的角度考虑的，而影响是从商业的角度考虑的。

所以当我们评估（风险）发生的可能性时，需要考虑哪些技术因素呢？以下所列的各项可以帮助你从何处入手：

- 技术和团队的复杂程度；
- 人员和培训的问题；
- 团队内部和团队之间的冲突；
- 供应商和卖方的合同问题；
- 就外包开发来讲，开发团队的地域分布问题；
- 之前遗留下的或是已建立的设计和技术与新的设计和技术的差异比较；
- 使用的工具和技术的质量，或是质量的缺失；
- 缺乏良好的管理上和技术上的领导；
- 时间、资源和管理的压力，尤其是存在财务上的惩罚制度时；
- 在开发生命周期中缺乏早期的测试或是质量保证活动；
- 项目中高频率的需求、设计和编码的变更；
- 高缺陷率；
- 复杂的接口和集成问题。

那么当我们评估影响时，需要考虑哪些商业因素呢？以下所列的各项可以帮助你从何处入手：

pin51.com 拼客发 资源分享

- 受影响特性（feature）的使用频率；
- 潜在的对企业形象的损害；
- 顾客和业务的损失；
- 潜在的对金融、生态环境或是社会造成的损失或不利；
- 民事或刑事法律的约束性；
- 失去经营许可证、执照等；
- 缺乏合理的解决办法；
- 失效和相关负面公众影响的可见性。

我们可以试图通过定量或者定性的方法确定风险的等级。定量分析风险时，我们可以对可能性和影响给出数值等级，可能性一般用百分比表示，影响一般用货币数量表示。如果我们把两者相乘，可以计算出风险发生的成本，在保险行业即被称为预期支出或预期损失。

在未来的软件工程中，说不定我们哪天就会将这一活动当作常规来做，一般地我们必须要定义出风险的等级。为什么呢？因为在进行质量风险分析时我们缺乏有效的统计数据。所以我们只可以将可能性定性地表示为很高、高、中、低或非常低，而不能用一种富有真正意义的方式来表达——发生的可能性是 90%、75%、50%、25%还是 10%。

但是，这并不是说定性的方法是低级的、无用的。事实上，根据我们平时工作中接触到的数据来讲，在大多数的项目中定量的方法并不合适。该方法貌似真实的精准度反而会误导利益相关者如何理解和管理风险。我发现，如果我接受数据的局限性而同时又使用适当的非正式的质量风险管理方法，得出的结果不仅非常有用，而且对于一个管理得当的测试过程至关重要。[2]

除非你的风险分析是基于大量的、统计过的、有效的风险数据，才有可能反映出相对准确的可能性和影响。换言之，是利益相关者持有的个人认知和观点决定了风险的等级。当然，这绝对没有错，我一点都没有谴责这一技术的意思。主要是项目经理、编程人员、用户、商务分析人员、设计人员和测试工程师通常有不同的认知，因此对每个风险项的风险等级会有不同的观点。我们需要在所有的这些不同的认知中，提炼出团队智慧的精华。

然而，利益相关者很有可能对风险的识别产生分歧。因而风险分析过程还应该包含一些方法使大家能达成共识。即便是在最差的情况下，如果我们无法达成共识，也要能够将这些分歧提交给某些管理层解决。否则，风险等级将是充满歧义和冲突的，那么对于风险减轻活动包括测试就不会有任何指导意义。

3.9.4 风险减轻或风险控制

在识别和评估风险过后，现在就需要控制这些风险。正如我之前提到过的，高级大纲将之称为风险减轻，可是这个定义并不是很正确。事实上，对风险控制通常有以下 4

2 我在 Pragmatic Software Testing 中讲到了关于风险的一些非正式的方法及主要的量化评估。另外，你也可以在 Rick Craig 写的 Systematic Software Testing 和 Erik van Veenendaal 写的 The Testing Practitioner 中找到相关的内容。

pin51.com 最新最全 拼吾爱

种选择：

- 减轻，我们采取一些预防措施来较少风险发生的可能性和/或风险发生产生的影响。
- 应急，采取一个或多个应急计划用以减少风险发生时产生的影响。
- 转移，当风险发生时，将风险产生的后果转交给另外的组织。
- 最后一个选择就是忽视或是接受风险及其产生的后果。

针对给定的风险项，选择上述一个或多个策略，都会有其收益、机会、成本，甚至潜在的其他相关风险。

基于风险的测试分析主要为测试团队，包括测试分析师创建风险减轻的机会，尤其是质量风险。基于风险的测试通过贯穿于整个软件开发生命周期的测试活动减轻质量风险。

请注意，在某些情况下，可以利用一些标准。我们已经看过一个这样的标准，美国联邦航空局 DO-178B 标准。后面将会看到另外一个这样的标准。

控制项目中的风险也是尤为重要的。对于测试分析师，我们尤其关注以下这些测试相关的项目风险：

- 测试环境和工具的准备就绪。
- 测试人员的可用性和合格性。
- 测试的低质量输入。
- 测试过程中的过多变更。
- 对于测试工作量（的管理）缺乏标准、规则和技术。

确保这些风险得到控制通常是测试经理的工作，而在这些方面缺乏充分的控制将会影响到测试分析师。

在基础级大纲中曾经讨论过一个观点，一个测试的基本准则，即及早进行测试和质量保证的准则，该准则强调预防潜在的风险。预防性测试是基于风险测试的一部分，它需要在开始测试执行之前采取措施尽量减轻风险，这些措施可能包括：及早准备测试件（testware），预先测试好测试环境，在各个测试级别正式开始前预测试产品的早期版本，坚持更严格的测试准入标准，确保需求和设计的可测试性，参与评审（包括对一些早期的项目活动的追溯），参与问题和变更管理，以及监控项目的进展和质量。

在预防性测试中，我们在整个软件生命周期一直都在采取质量风险控制的行动。测试分析师应该在合适的时机利用以下技术控制风险：

- 选择恰当的测试设计技术。
- 评审和审查（inspection）。
- 评审测试设计。
- 一些测试级别应具备一定程度的独立性。
- 在测试任务中使用最有经验的人。
- 选择合适确认测试和回归测试策略。

预防性测试策略表明质量风险能够，也应该，通过广泛的活动得以减轻，这些活动中很多都不是我们平时认为的“测试”。例如，如果需求写得不是很好，或许我们应实行

评审以改善其质量，而不是依赖于差的需求变成差的设计，最终变成差的、错误百出的代码之后进行的测试。

当然，测试并非对所有类型的质量风险都有效。在一些情况下，你可以预估到测试对降低风险的大体作用，以及某些测试技术对降低指定风险的作用。在测试效率很低的情况下，测试很难降低风险。例如，缺少注释或者使用非结构化的编程技术导致的可维护性的问题很难显现，至少在测试的初期不会显现出来。

一旦测试执行开始，我们将利用测试执行减轻质量风险。当发现缺陷时，测试人员通过提醒缺陷的存在和提供发布之前处理这些缺陷的机会来降低风险。当测试没有发现缺陷时，测试所降低的风险是通过确保在某些特定条件系统能够运行正常来体现的。我之前提到过，我们在基于风险的测试策略中通过风险的等级来决定测试的优先次序。这可以通过若干方法实现，这些方法的两个极端，即深度优先和广度优先。在深度优先方法中，所有最高风险的测试都在风险低的测试之前执行，测试按照严格的风险顺序执行；在广度优先的方法中，我们在指定的风险级别中，选定能够覆盖所有已识别的风险的测试样本，同时这些测试样本应该至少覆盖所有已识别的指定级别的风险一次。

当我们运行测试时，应该根据遗留下来的风险度量和记录我们的结果。在该领域测试覆盖率越高，遗留下来的风险越少。测试覆盖某风险领域越充分，遗留下来的风险越少。[3]当然，在进行基于风险的测试时，如果我们只是基于风险分析测试，这会导致一些盲点，所以我们需要利用预定义的测试规程以外的测试来确定我们是否遗漏了什么。

如果在测试执行过程中，需要减少在测试上花费的时间和工作量，我们可以以风险分析为基础。如果遗留下来的风险是可以接受的，我们可以缩短测试时间，减少测试工作量。请注意，通常这些未运行的测试不如已经运行的测试重要。如果我们缩减了下一步的测试，基于风险的测试的固有属性决定了遗留下来的风险将会转移到终端用户/客户、服务/技术支持人员或是运营/操作人员。

假设还有时间继续执行测试呢？在这种情况下，可以基于从当前测试获得的经验，将新的测试周期的风险分析及测试做出相应调整。首先，应该修订已有的测试风险分析。其次，对现存的测试和可能新加入的测试进行优先次序的重排。该如何决定我们是否需要调整风险分析呢？可以从以下几个主要因素开始：

- 全新的或是变更很大的产品风险。
- 测试中不稳定或是容易发现缺陷的领域。
- 风险，尤其是与修正缺陷关联的回归风险。
- 发现预料之外的缺陷群。
- 发现被遗漏的关键商业领域。

所以，如果有时间开始额外的测试周期，应该首先考虑修改质量风险分析。你还应该在每个项目里程碑阶段更新质量风险分析。

3 读者可以在我撰写的 Critical Testing Processes 以及本书的关于测试管理的姊妹篇中，找到关于基于风险记录测试结果的相关例子。

pin51.com 拼吾爱 最新最全源码分享下载

3.9.5 关于风险识别和评估结果的举例

在图 3-1 中，会看到一个关于质量风险分析文档的案例，这是从一个实际项目的研究中得到的。这个文档包括我们使用的方法，是按照失效模式和影响分析的方法步骤进行的。

1	失效模式和影响分析（质量风险分析）格式——按风险优先级数（RPN）排序											
8	原始 FMEA——按 RPN 排序											
9	系统功能或特性	潜在失效模式-质量风险	引发的潜在影响	是否关键	严重程度	失效潜在的起因	优先级	检测方法	检测难度	风险优先级	建议采取的行动	责任人/时间节点
10	碎片交换文档	分割碎片失败	安全缺口	是	1	程序错误	1	测试：调试 跟踪：代码评审	1	1	测试：调试 跟踪：代码评审	
11	碎片交换文档	大量碎片	数据丢失	是	1	程序错误	1	测试：调试 跟踪：代码评审	1	1	测试：调试 跟踪：代码评审	
12	压缩兼容性	数据损坏	数据丢失	是	1	程序错误	1	测试	2	2	测试	
13	压缩兼容性	系统挂起	数据丢失	是	1	程序错误	1	测试	2	2	测试	
14	压缩兼容性	切割碎片不当	数据丢失	是	1	程序错误	1	测试	2	2	测试	
15	互联网文档识别	识别不当	数据丢失	是	1	程序错误	1	测试：调试 跟踪：代码评审	2	2	测试：规则验证	
16	网络兼容性	网络碎片	数据丢失	是	1	程序错误	1	测试	2	2	测试选择/改进网络覆盖率	
17	删除文件名	FS 损坏	数据丢失	是	1	程序错误	1	测试：调试 跟踪：代码评审	2	2	测试：调试 跟踪：代码评审	
18	删除文件名	删除失败	安全缺口	是	1	程序错误	1	测试：调试 跟踪：代码评审	2	2	测试：调试 跟踪：代码评审	

图 3-1　一个关于使用 FMEA 的质量风险分析文档的案例

先从表的左边开始，先描述一个具体的功能，然后识别其失效模式和可能会产生的影响。影响程度决定了（各功能的）关键程度严重程度和优先级。我们列出可能的起因以使缺陷预防工作能够在需求、设计和实施阶段开展。

这之后，我们来看一下应对的检测方法，我们是期望该项目至少应该采用这些方法的。失效模式越是有可能在检测中被遗漏，则检测难度越大。我们是基于严重程度、优先级和检测难度来计算一个风险优先级数的。数值越低，代表风险的情况越差，严重程度、优先级和检测难度的数值范围都是 1～5，所以，风险优先级数的数值范围是 1～125。

这个特定的表格只显示了一些最高级别的风险项，因为我是按照风险优先数进行排序的。对于这些风险项，我们当然期望对其采取更多额外的检测和其他风险控制措施。你可以看到我们此时只确定了一些额外要采取的行动，但是还没有指派该行动的责任人。

在与一个风险项相关的测试行动中，我们期望当风险提高时，测试用例的数目、测

试数据的总数和测试覆盖率也应该随之提高。请注意我们可以允许覆盖一个风险项的任何一个测试规程继承该风险项的风险等级，也就是说通过该风险项的风险等级来定义测试规程的优先级。

3.9.6 软件生命周期中的基于风险的测试

我曾经提到过，基于风险的测试对风险的管理贯穿软件生命周期的始末。让我们一起看看，在一个测试项目中，我通常运用的方法。

在测试计划期间，首先进行风险管理。我通常在项目的早期，比较理想的是在需求规格说明初稿完成时，便开始进行质量风险分析。根据质量风险分析，我得出一份评估报告，用以与项目的利益相关者及管理层协商，并取得他们的认可。

一旦项目的利益相关者和管理层同意我的这份评估报告，我就会写一份测试计划。该计划会定义测试工作量，并基于质量风险的级别定义出测试的顺序。还会对影响测试的项目风险做出计划。

在测试控制期间，我将在项目的全过程中定期调整风险分析。这将会引发测试覆盖率的增加或减少；移除、延迟或是提前测试的优先级；或是其他的一些类似的行为。

在测试分析和设计期间，我与测试团队一同根据风险分配测试开发和执行的工作量。因为我希望按照风险来记录测试结果，以确保质量风险的可追溯性。

在测试实现和执行期间，我按照风险的等级对测试规程排序。我还要确保测试团队使用探索性测试和其他互动技术来检测未曾预料到的高风险领域。

在出口准则评估和报告期间，我与我的测试团队一起根据风险来衡量测试的覆盖率。在报告测试结果（准备发布）时，我们不仅要讨论测试用例的执行情况和发现的缺陷，还要描述遗留风险。

3.9.7 （具备）风险意识的测试标准

如在第2章中所看到的，美国联邦航空局DO-178B标准在白盒测试的代码覆盖率对失效的潜在影响的分析基础上对测试内容进行扩展，这就使得DO-178B成为一个具备风险意识的测试标准。

如高级大纲所述，另一个关于如何将风险管理，包括质量风险管理，应用到复杂和/或安全关键系统工程的是ISO/IEC 61508。通过该标准的标题《电子、电气、可编程电子器件的安全相关系统的功能安全》，可以发现该标准适用于通过安全相关的嵌入式软件。

该标准非常关注于风险，因此风险分析是必需的。它将两个重要因素视为决定风险等级的要素，即可能性和影响。在一个项目中，特别是在涉及通过电子（器件）、电气设备或软件来增强功能的系统的项目中，我们希望将遗留下来的风险的等级减少到一个能够承受的程度。

该标准反映了风险的本质。它认可我们是不可能对整个系统，甚至对一个单独的风险项，做到零风险控制的。也就是说，我们需要从一开始就考虑构建质量，尤其是安全，而不是在最后才考虑。因此我们必须采取类似于需求、设计和代码评审等缺陷预防措施。

该标准还坚持我们应该了解什么构成可承受和不可承受风险，以及采取行动减少不

pin51.com 最新最全软件资源 拼吾爱

可承受风险。当这些行动属于测试范畴，我们就必须将其文档化，包括软件安全确认计划、软件测试规格说明、软件测试结果、软件安全确认和验证报告以及软件功能安全报告。

该标准还强调了笔者偏见的问题。如在基础级大纲中讨论过的一样，这是一个关于自测试的问题，在测试自己作品的时候，作者往往会出于对自己作品的偏好引入测试盲点或者错误的假设。所以标准要求测试人员的独立性，尤其强调在进行任何与安全相关的测试时，要保证测试的独立性。由于当被开发系统的可测试性较高时，测试才是最有效的方法，所以系统的可测试性也是要求之一。

该标准还提到了一个关于安全集成等级（Safety Integrity Level，SIL）的概念，是根据对于某个特定组件或子系统失效的可能性得出的。安全集成等级会影响到相当数量的关于风险的决策，包括对测试和质量保证技术的选择。

一些技术我们在本书中有所涉及，例如，各种功能和黑盒测试设计技术，许多技术我会在该书的关于高级测试技术分析师的姊妹篇中讲到，包括概率测试、动态分析、数据记录和分析、性能测试、接口测试、静态分析和复杂度度量。此外，因为通过彻底的覆盖率，包括在回归测试期间，以减少缺陷遗漏的可能性是非常重要的，该标准要求使用适用的自动化测试工具（以达到彻底的代码覆盖率）。

再次强调，根据安全集成等级，该标准要求应用各种不同的测试级别。包括组件测试、集成测试、软硬件集成测试、安全需求测试和系统测试。

如果要应用一个（安全集成）测试等级，该标准表明该测试级别应该文档化，并且可以独立验证。换言之，该标准要求对这些测试行为可以进行审计或是外部评审。此外，秉承“对监管者的监管（guarding the guards）”的思想，该标准要求在验证满足测试条件的数据集成的同时对测试用例、测试规程和测试结果进行评审。

该标准还要求使用结构化测试设计技术，并根据安全集成级别满足相应的结构化覆盖率需求（这一点与 DO-178B 标准类似）。因为旨在系统的安全性方面获取较高的信心，标准不止一次，甚至是多次地对于多个测试级别，要求 100%需求覆盖率。再次强调，测试覆盖率的级别要求取决于安全集成级别。

以上讲到的内容，好像有些过多了，尤其当你来自一个凡事不必走严格流程的环境里。不过，下次当你步入两片能活动的金属（例如电梯门）之间时，问问自己你准备在控制它们运动的软件中遗留多少风险。

3.10 基于风险的测试练习 1

阅读附录 B 中的 HELLOCARMS 系统需求规格说明，该系统是从一个 RBCS 帮助测试的真实项目中衍生出的一个模拟项目。

如果你在一间教室里做该练习，请加入到任何一个 3～5 人小组团队合作做该练习。如果你是单独做练习的，那么需要自己完成为该项目进行质量风险分析的工作。因为本书主要关注于域测试，而且仅对功能的质量风险进行识别和评估，请使用图 3-2 和表 3-1 中所示的模板。

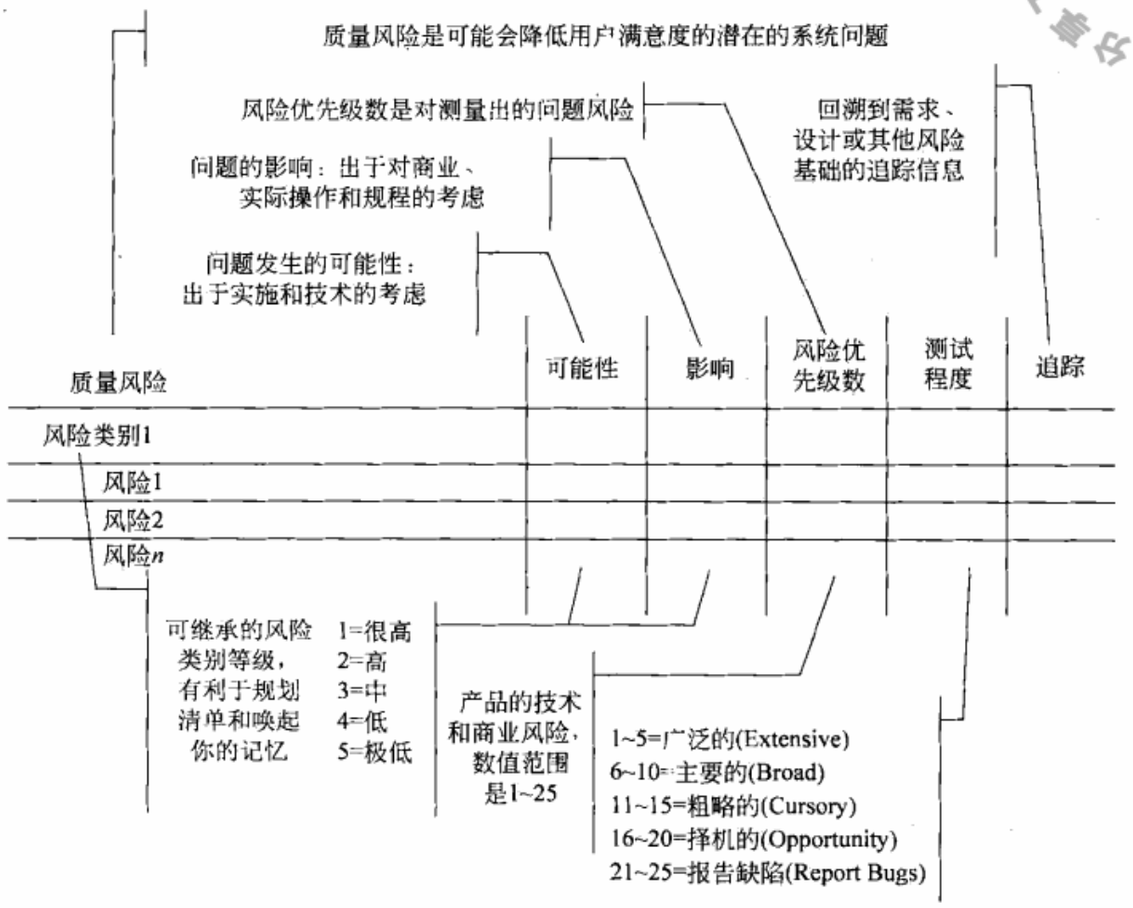


图 3-2 带注解的关于非正式质量风险分析的模板

表 3-1 显示待解决的质量风险类别的功能质量风险分析模板

NO.	质 量 风 险	可能性	影响	风险优先级数	测试程度	追踪
1.1.000	功能性：适用性					
1.1.001	[将与适用性相关的功能风险写入该栏]					
1.2.000	功能性：准确性					
1.2.001	[将与准确性相关的功能风险写入该栏]					
1.3.000	功能性：互操作性					
1.3.001	[将与互操作性相关的功能风险写入该栏]					
1.4.000	功能性：安全性					
1.4.001	[将与安全性相关的功能风险写入该栏]					
1.5.000	功能性：合规性					
1.5.001	[将与合规性相关的功能风险写入该栏]					

为了保证时间安排的合理性，我建议先花 30～45 分钟识别质量风险，然后再花 15～30 分钟对每个风险级别进行评估。如果你是在一个教室中分组做该练习，一旦分析环节各自的风险结束，即请对结果进行讨论。

pin51.com 拼吾爱 最新最全源分享下载

3.11 基于风险的测试的练习解答1

可以在表3-2中看到我对该练习的解决方案。在这张表之后是两个副产品的清单，一个是在分析期间发现的项目风险的清单，另外一个是在分析过程中发现的需求文档中缺陷的清单。

表3-2 HELLOCARMS系统的功能质量风险分析

NO.	质量风险	可能性	影响	风险优先级数	测试内容	追踪
1.1.000	功能性：适用性					
1.1.001	拒绝所有房屋净值贷款的申请	5	1	5	广泛的	010-010-010
1.1.002	拒绝所有房屋净值信用贷款的申请	5	1	5	广泛的	010-010-010
1.1.003	拒绝所有反向住房按揭贷款的申请	5	1	5	广泛的	010-010-010
1.1.004	无法正确处理某些房屋净值贷款的申请	3	1	3	广泛的	010-010-190
1.1.005	无法正确处理某些房屋净值信用额度的申请	3	2	6	主要的	010-010-200
1.1.006	无法正确处理某些反向住房按揭贷款的申请	3	3	9	主要的	010-010-210
1.1.007	无法正确处理某些组合产品（例如，房屋净值和信用卡）的申请	3	4	12	粗略的	010-010-220
1.1.008	无法正确处理某些已有的住房按揭贷款	3	5	15	粗略的	010-010-230
1.1.009	无法正确处理已批准的申请	3	4	12	粗略的	010-010-240
1.1.010	无法对所有填空域和页面产生脚本	4	2	8	主要的	010-010-020
1.1.011	无法采集顾客原始数据	3	2	6	主要的	010-010-030
1.1.012	顾客原始数据类别定义不当	2	2	4	广泛的	010-010-030
1.1.013	输入栏接受无效数据	1	1	1	广泛的	010-010-040
1.1.014	现存的交易信用不能正确显示或处理	3	1	3	广泛的	010-010-050 010-010-100
1.1.015	交易信用的细节不能传到LoDoPS	2	1	2	广泛的	010-010-050 010-010-100
1.1.016	收入债务比率包括了应付贷款	4	3	12	粗略的	010-010-010
1.1.017	无法恢复不完整/被中断的申请	3	2	6	主要的	010-010-060
1.1.018	未查询申请人当前与Globobank的关系	5	2	10	主要的	010-010-070
1.1.019	申请人与Globobank的关系没有发送到GLADS	3	2	6	主要的	010-010-070
1.1.020	贷款状态信息在初始化之后丢失	3	2	6	主要的	010-010-080
1.1.021	不能彻底取消或中止一个申请（例如：必须要关闭浏览器）	4	3	12	粗略的	010-010-090
1.1.022	不能按照顾客ID恢复现存的申请	4	4	16	择机的	010-010-120
1.1.023	没有将超过$500 000的贷款提交审批	4	1	4	广泛的	010-010-130

续表

NO.	质 量 风 险	可能性	影响	风险优先级数	测试内容	追踪
1.1.024	自动拒绝超过$500 000 的贷款	3	1	3	广泛的	010-010-130
1.1.025	没有将资产价值超过$1 000 000 的申请提交审批	4	2	8	主要的	010-010-140
1.1.026	自动拒绝资产价值超过$1 000 000 的申请	3	2	6	广泛的	010-010-140
1.1.027	在承诺提供服务的地区无法提供本地电话销售支持	1	2	2	广泛的	010-010-150
1.1.028	在承诺提供服务的地区无法提供长途电话销售支持	1	2	2	广泛的	010-010-150
1.1.029	不能支持对经纪人和其他商务伙伴的品牌宣传	4	2	8	主要的	010-010-160
1.1.030	未经训练的用户（例如，终端客户）不能通过互联网访问应用程序	4	3	12	粗略的	010-010-170
1.1.031	不能支持零售业务银行分支机构的产品操作	4	4	16	择机的	010-010-180
1.1.032	不能支持灵活的价格实施方案	4	5	20	择机的	010-010-250
1.2.000	功能性：准确性					
1.2.001	顾客收到他们无资格获取的（金融）产品	3	1	3	广泛的	010-020-010
1.2.002	顾客不能收到他们有资格获取的产品	4	1	4	广泛的	010-020-010
1.2.003	申请的决策机制与全球银行信用政策不符	3	1	3	广泛的	010-020-010
1.2.004	根据信用记分、贷款成数和负债收入来计算的基于风险的价格的计算错误	4	1	4	广泛的	010-020-020
1.2.005	新的贷款支付情况未包含在信用记分中	3	2	6	主要的	010-020-030
1.2.006	不能正确计算附加价格	3	3	9	主要的	010-020-040
1.2.007	不能正确处理政府的退休收入	3	1	3	广泛的	010-020-050
1.2.008	不能获得额外收入的持续时间	4	3	12	粗略的	010-020-060
1.3.000	功能性：互操作性					
1.3.001	HELLOCARMS 无法从 GloboRainBQW 获取信息	2	2	4	广泛的	010-030-010
1.3.002	HELLOCARMS 和信用评分主机 Scoring Mainframe 拒绝联合申请	3	1	3	广泛的	010-030-020
1.3.003	HELLOCARMS 和 Scoring Mainframe 不能处理/解决联合申请中的重复信息	1	2	2	广泛的	010-030-030
1.3.004	HELLOCARMS 的交易记录与 LoDoPS 通讯失败	3	1	3	广泛的	010-030-040 010-030-070
1.3.005	来自 LoDoPS 的关于贷款状态的信息转移到 HELLOCARMS 时丢失或中断	5	2	10	主要的	010-030-050 010-030-140
1.3.006	如果 Scoring Mainframe 显示有未清偿的破产或丧失抵押品赎回权，HELLOCARMS 不能继续	4	3	12	粗略的	010-030-060

pin51.com 最新最全源码 拼吾爱

续表

NO.	质量风险	可能性	影响	风险优先级数	测试内容	追踪
1.3.007	HELLOCARMS与LoDoPS交互政府退休收入信息时失败	5	1	5	广泛的	010-030-080
1.3.008	HELLOCARMS 申请信息未能正确传递给Scoring Mainframe	3	1	3	广泛的	010-030-090
1.3.009	HELLOCARMS 不能正确接收来自 Scoring Mainframe 的信息	3	1	3	广泛的	010-030-100
1.3.010	Scoring Mainframe 中决策请求未按照要求排队	4	2	8	主要的	010-030-110
1.3.011	通过初审并得到顾客认可的贷款不能传到LoDoPS	5	2	10	主要的	010-030-120
1.3.012	被拒绝的申请不能传到LoDoPS	5	2	10	主要的	010-030-130
1.3.013	LoDoPS 中的贷款信息变更不能被传回HELLOCARMS	5	2	10	主要的	010-030-140
1.3.014	不支持计算机电话语音的集成服务	3	5	15	粗略的	010-030-150
1.3.015	申请人与Globobank的关系不能传给GLADS	4	3	12	主要的	010-010-070
1.4.000	功能性：安全性					
1.4.001	不支持已达成一致的安全性需求	1	2	2	广泛的	010-040-010
1.4.002	“由……创建”和“最后一次由……变更”的审计追踪信息丢失	3	1	3	广泛的	010-040-020
1.4.003	外包的电话销售人员会看到实际的信用记分和其他（不应该获知的）特权信息	3	2	6	主要的	010-040-030
1.4.004	互联网应用程序可能遭到来自互联网的恶意攻击的不安全性	1	2	2	广泛的	010-040-040
1.4.005	互联网应用程序可能遭到来自非互联网的恶意攻击的不安全性	3	2	6	主要的	010-040-040
1.4.006	不允许通过互联网匿名浏览	2	4	8	主要的	010-040-050
1.4.007	欺诈侦测过度宽松	3	1	3	广泛的	010-040-060
1.4.008	欺诈侦测过度严格	3	1	3	广泛的	010-040-060
1.5.000	功能性：合规性					
1.5.001	[将与合规性相关的功能风险写入该栏]					

在第一轮质量风险分析中，我审查每个功能需求，并识别出一个或多个与之相关的风险项。假定需求的优先级很好地表达了风险的影响，所以我在风险分析中采用了该优先级（定义风险潜在的影响）。即便使用了这些快捷的办法，我仍然花了一个小时的时间完成审查的过程。

3.11.1 项目风险的副产品

在准备质量风险分析文档的过程中，我发现以下存在于需求规格说明中的项目风险。

由于缺乏清晰的安全需求（详见 010-040-010），很有可能造成支撑该项目进度的必要基础架构无法按计划到位。

3.11.2 需求缺陷的副产品

在准备质量风险分析文档的过程中，我发现以下存在于需求规格说明中的缺陷：

1. 对于 010-010-150，支持哪些州、省及国家？
2. 对于 010-010-150，词组“所有州（all support States）”应该是“所有的州（all supported States）”。
3. 对于 010-020-020，词组“收入深度（dept-to-income）”应为“收入负债率（debt-to-income）”。
4. 对于一些需求，ID 号重复，例如 010-030-140。
5. 不是所有的需求都定义了优先级，例如 010-030-150。
6. 对于 010-040-010，这里“支持商定的安全要求”是什么？
7. 需求规格说明里没有直接提到关于规范一致性的需求，尽管有很多适用于银行系统的规范。

3.12 基于风险的测试练习 2

通过对 HELLOCARMS 进行质量风险分析，可以概括出一组用于 HELLOCARMS 系统集成测试和系统测试的功能测试套件。特别需要注意的是，系统集成测试级别的整体目标，在于找到 HELLOCARMS 应用程序在数据处理中心与其他应用程序相互协作的能力中存在的缺陷并建立相关质量保障。同时，系统测试级别的整体目标，则在于找到 HELLOCARMS 应用程序提供必要（系统功能、性能的）能力时存在的缺陷并建立相关质量保障。

最后，要阐明在测试执行阶段，如何为测试套件中的测试用例排序的原则。

再说一次，如果是在教室中做这个练习，一旦小组完成各自的测试套件设计和原则的陈述，就应该开始对结果进行组内讨论。

3.13 基于风险的测试练习解答 2

根据所进行的质量风险分析，我创建了如下所示的系统集成测试和系统测试套件。

系统集成测试套件

- HELLOCARMS/Scoring Mainframe 之间的接口。
- HELLOCARMS/LoDoPS 之间的接口。
- HELLOCARMS/GLADS 之间的接口。
- HELLOCARMS/GloboRainBQW 之间的接口。

系统测试套件

- 基本功能。

pin51.com 最新最全源代码 拼吾爱

- 不受欢迎的申请人。
- 端对端交易。
- 错误的处理和恢复。
- 安全。
- 一致性。

3.13.1 测试用例排序原则

由于该应用程序包括了范围很广的功能，需要处理许多不同的情况，我决定采用广度优先的测试顺序原则，如下面所列的测试套件。在广度优先的方法中，我们在指定的风险级别中，选定能够覆盖所有已识别的风险的测试样本，这些测试样本应该至少覆盖所有已识别的指定级别的风险一次。

所以，在开始测试执行之前，我们利用由质量风险分析得到的可追溯的信息，如果测试规程（包含）覆盖高等级风险测试项（的测试用例），则该测试规程将继承这个高风险等级。然后将所有测试用例分为如下 3 个子集：

- 高优先级：测试规程的一个子集，占计划测试工作总量的 1/3。50%的测试规程的风险优先级数为 1~5；35%的测试规程的风险优先级数为 6~10；15%的测试规程的风险优先级数为 11~15。
- 中优先级：测试规程的一个子集，占计划测试工作总量的 1/3。35%的测试规程的风险优先级数为 1~5；50%的测试规程的风险优先级数为 6~10；15%的测试规程的风险优先级数为 11~15。
- 低优先级：测试规程的一个子集，占计划测试工作总量的 1/3。15%的测试规程的风险优先级数为 1~5；15%的测试规程的风险优先级数为 6~10；70%的测试规程的风险优先级数为 11~15。

之后，将会迎来一个高优先级测试周期，一个中优先级测试周期和一个低优先级测试周期。在每个测试执行周期期间，我们按照严格的风险优先级数顺序执行子集中的测试用例。

这种排序方法需要根据测试版本发布时间表和总测试集的大小灵活调整。例如，如果完成总测试集计划持续六周，而测试版本发布频率是每周一次，我们会有 3 个为期两周的周期，每个周期应接收两个测试版本。如果完成总测试集计划持续八周，则 4 个周期而不是 3 个会更合理一些。

您可能会注意到，该排序方法是以假设优先级数为 1~5、6~10 和 11~15 的测试规程数量相对平衡为前提的，如果该前提不成立的话，则计划需要有所调整。不过，无论测试规程的风险优先级数实际上是如何混合的，计划的基本意图都是可以实现的。

除实现计划的测试工作量之外，还有两种额外的测试应该予以考虑并纳入工作计划：

- 确认测试：当我们收到计划内或计划外的测试版本时，对已修正的缺陷的测试。
- 反应式测试：使用探索性测试、Whittaker 攻击、缺陷分类、Hendrickson 的缺陷捕猎（bug hunting）技术，以及其他的非系统化的测试技术（这些技术在基础级大纲中已经涉及到，会在本书后面部分详细讲述）。一些反应式测试主张将等级

为“择机的”或更低的风险项，同在风险分析期间未能识别的风险项一起，充当质量风险分析的制衡原则。

每个测试周期大约有 60%～70%会进行计划性测试，约 30%～40%的时间用来做确认测试和反应式测试。

3.14 失效模式和影响分析

学习目标：

能回想起该部分内容即可。

本章节中的概念主要适用于测试经理，对测试分析师关于这部分内容的学习目标不做要求。所以，如果您正在为准备认证考试而学习，那么，可以通过复习高级大纲中的第 3 章回想和熟悉这部分内容。

ISTQB 术语

基于会话的测试管理（Session-based test management）：在术语表中无定义。

3.14.1 测试管理的相关问题

学习目标：

能回想起该部分内容即可。

本章节中的概念主要适用于测试经理，对测试分析师关于这部分内容的学习目标不做要求。所以，如果您正在为准备认证考试而学习的话，那么，可以通过复习高级大纲中的第 3 章回想和熟悉这部分内容。

3.15 认证考试模拟题

在每个章节的最后，您可以试着做几道认证考试模拟题以加强对材料的记忆和理解，准备 ISTQB 高阶测试分析师的考试。

1. 在 IEEE 829 的众多模板中，其中在项目生命周期早期使用的文档中的一个章节，与在之后的项目生命周期中使用的多个文档中的类似章节相关联。例如，一个符合 IEEE 829 的测试计划包括一个叫做“待测试特性”的章节，那么你肯定希望在这个测试计划的该章节中列出的每个特性都能被其他的测试设计规格文档中的“待测试特性”章节所包含。那么以下哪种说法最好地表达了 IEEE 829 模板的另外一个相似关系？

A 测试用例规格说明对测试规程规格说明好比测试计划对测试设计规格说明

B 测试规程规格说明对测试用例规格说明好比测试计划对测试设计规格说明

C 测试用例规格说明对测试项传递报告好比测试计划对测试设计规格说明

pin51.com 最新最全 拼吾爱

D 测试计划对测试项传递报告好比测试计划对测试设计规格说明

2. 假设你是一名测试分析师，正在负责一个银行项目，将现有的自动提款机升级为允许用户用该银行支持的信用卡预支现金。在测试计划中的一个出口准则中要求从所有支持的信用卡中成功提现至少500欧元，并将这一过程进行记录。正确的支持的信用卡清单为，American Express、Visa、Japan Credit Card、Eurocard和MasterCard。

在风险分析期间，你识别的多个质量风险项中，你将按照对风险的认知从最高到最低排列如下：

- Visa或MasterCard不能提取现金。
- American Express或Eurocard不能提取现金。
- Japan Credit Bank不能提取现金。
- Visa或MasterCard提取现金限额不正确。
- American Express或Eurocard提取现金限额不正确。
- Japan Credit Bank提取现金限额不正确。

假设，如下所列的测试用例中，每个测试用例要求恰好一个小时进行测试分析。那么，下面哪个由分号分隔的顺序测试用例清单覆盖了这些风险等级，并使测试工作量最小化？

A 测试American Express和Eurocard的最小提现金额；测试Visa和MasterCard的最小提现金额；测试Japan Credit Bank的最小提现金额

B 测试Visa和MasterCard的最小和最大提现金额；测试American Express和Eurocard的最小和最大提现金额；测试Japan Credit Bank的最小和最大提现金额

C 测试Visa和MasterCard的最小提现金额；测试American Express和Eurocard的最小提现金额；测试Japan Credit Bank的最小提现金额；测试Visa和MasterCard的最大提现金额；测试American Express和Eurocard的最大提现金额；测试Japan Credit Bank的最大提现金额

D 测试American Express和Eurocard的最大提现金额；测试Visa和MasterCard的最大提现金额；测试Japan Credit Bank的最大提现金额

3. 假设在一个项目进行期间，当前你只有如下3个文档或记录可用：

I 产品重要质量特性的检查清单；

II 关于在哪些区域发现了缺陷以及这些缺陷的严重程度的类似历史项目资料；

III 到目前为止本项目的测试执行阶段中已发现的子系统的缺陷的柏拉图分析。

上述文档或记录分别最适合应用于风险管理中的哪个活动？

A I：风险识别；II：风险评估；III：风险控制

B I：风险控制；II：风险评估；III：风险识别

C I：风险识别；II：风险控制；III：风险评估

D I：风险评估；II：风险控制；III：风险识别

pin51.com 拼吾爱

第4章 测试技术

> “在这件事上，您得懂一点空气动力学的基本知识。”
>
> ——The Gyro Captain
>
> “闭嘴，闭嘴！”
>
> ——“Mad” Max Rockatansky
>
> 在 Terry Hays 和 George Miller 的“战士之路（The Road Warrior）”剧本里，Max 礼貌地拒绝了一个 gyrocopter 领航员关于航空工程“精益生产（just-in-time）”的课程。

ISTQB 高级大纲的第 4 章讲的是测试技术方面的知识。为了让文章的内容更有条理，在该章节中使用了一种层次分类系统，将内容细分成各个小节。为了方便读者，它沿用了 ISTQB 基础级大纲中对测试技术的分类方法，即基于规格说明的测试技术、基于结构的测试技术和基于经验的测试技术，也在每个类别中增补了一些测试技术，并对所有覆盖技术进行了更加深入的阐述。该章节中也讨论了静态分析和动态分析。[1]

该章节包含了 6 个小节：

1. 概述。
2. 基于规格说明的测试技术。
3. 基于结构的测试技术。
4. 基于经验和缺陷的测试技术。
5. 静态分析。
6. 动态分析。

下面将详细讲解各节的内容以及它们和测试分析的关系。

4.1 概　　述

学习目标：

能回想起该部分内容即可。

1　您可能注意到此处的组织结构与基础级大纲中的组织结构有一些不同。在基础级大纲中，第 3 章静态技术讲解了静态分析技术，而对于动态分析技术，只是在第 6 章工具这一章节中稍许提到一点。

pin51.com 最新最全 拼吾爱

在本章节以及后续的两个章节中，我们将讲解一些测试技术。在此之前我会先给出这些测试技术的分类结构，然后指出哪些测试技术是要求测试分析师掌握的，哪些是要求测试技术分析师掌握的。

图 4-1 中将测试技术第一级分解为两个部分，静态测试技术和动态测试技术。在基础级大纲的定义中，静态测试技术指的是无需运行测试对象的测试技术，而动态测试技术是指需要运行测试对象的测试技术。

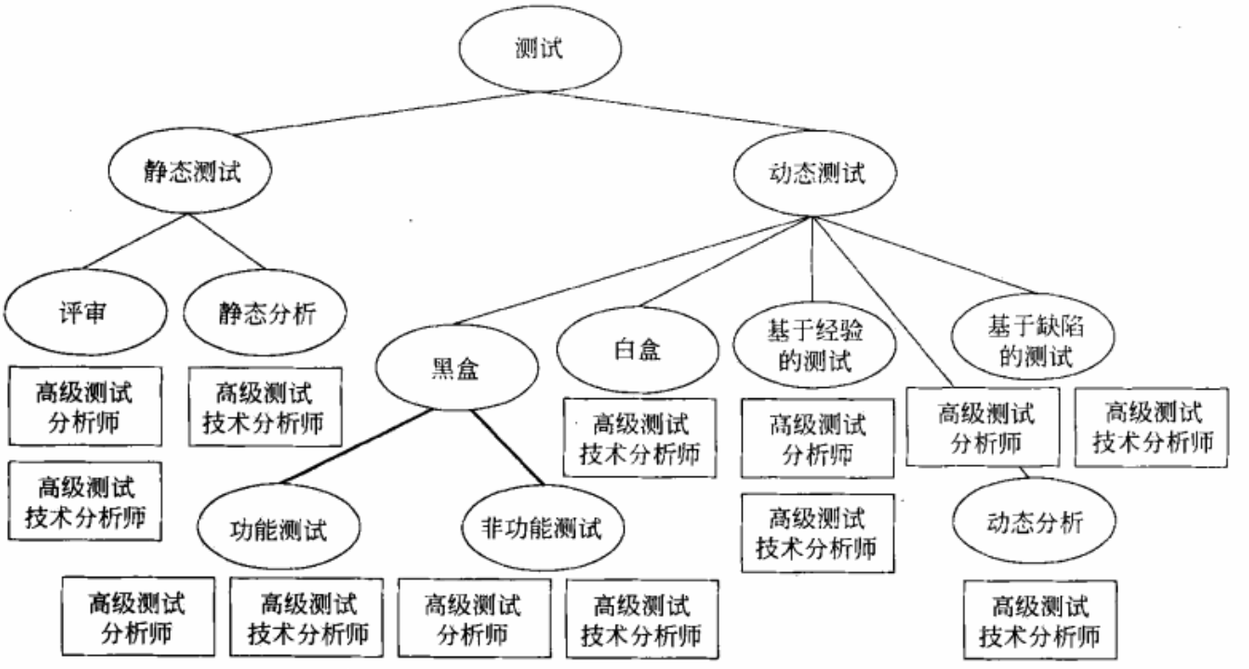


图 4-1　高级大纲中使用的测试技术分类系统

在高级大纲中，静态测试技术被分为评审和静态分析。评审是指那些以人为主要手段对测试项进行检查并寻找缺陷的方法，而静态分析则依靠测试工具来查找缺陷。

在高级大纲中，动态测试技术主要包含 5 种类型：

- **黑盒测试技术**（也叫基于规格说明或基于行为的测试技术）：我们基于已定义的系统行为进行测试。在高级大纲中，根据 ISO 9126 标准中的定义，将黑盒测试技术进一步分为功能和非功能测试技术。有一种很方便的方法能区分功能测试和非功能测试，即功能测试描述系统应该做些什么，非功能测试描述系统做得怎么样。
- **白盒测试技术**（也叫基于结构的测试技术）：我们基于系统的构建方法进行测试。
- **基于经验的测试技术**：我们基于我们的技能、直觉，以及在其他类似系统或技术中获得的经验进行测试。
- **动态分析**：在系统运行过程中，我们通过在代码中植入测量工具进行测试。
- **基于缺陷的测试技术**：在测试设计过程中根据我们对要测试的缺陷类型的理解，系统化地分析这种缺陷以生成测试用例。

在分类过程中，我们总希望能对各种技术明确地分类。如果您熟悉生物学，您应该记得各种生物，从疱疹病毒到树木再到黑猩猩，根据其所在的大的生物分类，都有一个

唯一的拉丁名字。然而，对于测试而言，这些测试技术都是互补的。无论您在进行什么级别的测试，对于任何测试活动您都应该使用尽可能多的恰当的技术进行测试。

在图 4-1 中分类树的叶节点的下方，您会看到一个或两个方格。标识为 ATA 的方格是指在高级测试分析师会用到的技术，我会在本书中详细讲解。标识为 ATTA 的方格是指高级测试技术分析师会用到的技术，我会在本书的姊妹篇《高级测试技术分析师》中对其进行讲解。绝大部分的测试技术我在两本书中都会提到，不过这并不意味着测试分析师和测试技术分析师使用同样的测试技术。

请注意我们仅将测试技术按照常规应用情况进行了角色分类，这可能与您所在的组织使用的分类有所不同。您可能身为测试分析师，却要承担动态分析的工作，您也有可能身为测试技术分析师，却不曾承担过任何静态分析的工作。

我想您应该已经明白在后续的章节中我们将学习哪些内容。虽然还有一些其他章节，如事件管理、测试团队和测试工具，但图 4-1 中的测试技术构成了本书的核心，覆盖了约 75%的篇幅。我会在本章讲解除了非功能测试之外的所有动态测试技术，在下一章节讲解非功能测试技术，在随后的章节再讲解评审技术。

您可以复制一份图 4-1 放在手边，以便您在本章和接下来的两个章节的学习过程中，能明确您正在学习哪个部分。

4.2 基于规格说明的测试技术

学习目标:

（K2）列出每个特定的基于规格说明测试技术如何发现典型缺陷的例子，并提供相应的测试覆盖率准则。

（K3）使用下面列出的测试用例设计技术，对给定的软件模型编写测试用例（需要满足给定模型中要求的测试覆盖）：等价类划分、边界值分析、决策表、状态转换测试、分类树方法、结对测试、用例。

（K4）分析一个系统或者它的需求规格说明，选择合适的基于规格说明的技术来满足特定的目标，并按照 IEEE 829 标准，列出功能测试用例、领域测试用例和测试规程的主要内容。

在我们深入到每种测试技术的细节之前，先对基于规格说明的测试技术进行概要的了解。

在基于规格说明的测试中，我们通过分析测试依据（test basis）设计并选择要执行的测试。测试依据是那些文档，它直接或间接地告诉我们被测组件或系统应该如何运行、应该做什么、做成什么样。我们根据这些文档进行测试。因此我们将它们称为测试依据。

在此有必要将测试依据和测试准则（test oracle）进行区分，它们有关联但并不完全相同。测试准则是任何我们用来定义（正确的）期望输出的事物，用以和被测组件和系统的实际输出进行比较（判断测试是否通过）。任何可以作为测试依据的文档同时也可能是一份测试准则。然而，测试准则还可以是现存的系统，比如一个正在被更换中的遗留

pin51.com 最新最全源码分享 拼吾爱

系统或竞争对手的系统。测试准则也可以是某人具备的专业知识。但我们不能将代码当成测试准则，因为那样的话我们只是在测试编译器是否能正常工作。

对于下一节介绍的基于结构的测试技术，系统的内部结构是测试依据，而不是测试准则。然而，对于基于规格说明的测试技术而言，至少在理论上，我们不需要考虑其内部结构。

在基于规格说明的测试技术中，除了关注系统的行为而不是结构这个共同点外，还有什么其他共同点呢？嗯，无论正式与否，还有所谓的模型。模型可以指一幅图或一张表格。我们可以根据模型系统地设计测试用例。通常每种测试技术会包含一系列的覆盖准则，以明确如何充分利用该模型提供的信息进行测试用例设计。您应该牢记的是，即使测试已经满足了某种测试技术所规定的覆盖率准则，这并不能说明测试已经完整或穷尽了。这只能说明您已经使用该种测试技术并依据该模型进行了有效的测试。

某些测试技术特别适合于查找一些特定类型的缺陷。伯利兹-贝哲（Boris Beizer）在他关于测试设计的书中将这种现象称为“缺陷假设”。他的意思是，如果您假定系统可能存在某种类型的缺陷，那么您可以根据假设选择测试技术（来设计测试）。这与基于缺陷的测试技术很类似，我们将在本章的后续章节中详细描述这种技术。[2]

ISTQB 术语

基于需求的测试：根据需求推导测试目标和测试条件以设计测试用例的方法。例如，执行特定功能的测试或探测诸如可靠性和可用性等非功能性属性的测试。

规格说明：说明组件/系统的需求、设计、行为或其他特征的文档，常常还包括判断是否满足这些条款的方法。理想情况下，文档是以全面、精确、可验证的方式进行说明的。

黑盒测试：不考虑组件或系统内部结构的功能或非功能测试。

基于规格说明的技术（或黑盒测试设计技术）：基于系统功能或非功能规格说明书来设计或者选择测试用例的技术，不涉及软件内部结构。

测试依据：能够从中推断出组件/系统需求的所有文档。测试用例是基于这些文档的。只能通过正式的修正过程来修正的文档称为固定测试依据。

测试准则：在测试时确定预期结果与实际结果进行比较的源。一个准则可能是现有的系统（用作基准），一份用户手册，或者是个人的专业知识，但不可以是代码。

通常，基于规格说明的测试是指基于需求文档的测试。需求规格说明描述了系统的行为，尤其是功能性行为。（由于需求描述中低估非功能需求而影响系统质量是个相对独立的话题，在此我们仅接受该观点，但不做详细讨论。）所以我们可以根据需求中描述的系统行为进行建模，然后再根据模型设计测试用例。

2 对于该主题，伯利兹-贝哲的书中包含了黑盒测试技术（也许这是与测试分析师最相关的部分）、软件系统测试与质量保证（与测试分析师、测试技术分析师、测试经理相关），以及软件测试技术（也许这是与测试技术分析师最相关的部分）。

pin51.com 最新最全源码资料下载

4.2.1 等价类划分

我们从最基础的基于规格说明的测试设计技术开始，即等价类划分。概念上讲，等价类划分就是测试各种不同的组，根据规格说明系统应该对于同一组中的每个值，都会以相同的方式处理，表现出相似的行为。

等价类划分技术的使用基础是一个能够识别等价类的模型，图形模型或数学模型，根据模型对输入值、输出值、内部值、时间关系、计算或其他任何适合的项进行等价类划分。之所以说这些类或划分是等价的，是因为系统应该以同样的方式处理它们。我们将某些类称为有效等价类，它们代表了有效的情况，因此系统对它们进行正常处理。我们将其他类称为无效等价类，它们代表了无效的情况，系统会拒绝处理或要求用户来改正操作或进行异常处理。

一旦我们识别了等价类就可以根据划分结果开始设计测试用例。通常我们会同时处理多个等价类划分。例如，同一界面上的每个输入域都有它自己的一组有效和无效等价类。因此我们可以从每个有效的等价类中各选一个有效值组成一组有效测试用例。重复这个过程，直到每个等价划分中的每个有效类都通过至少一个测试用例进行了测试。

接下来，我们可以开始设计无效测试用例。对于每个等价类划分，从中选取一个无效类，然后从其他所有等价类划分中各选一个有效类。之所以不在一个测试用例中将多个无效类组合在一起测试，是因为一个无效值可能会掩盖对另一个无效值的异常处理。每一个等价类划分中的任何无效类都应该被至少一个无效测试（用例）覆盖到。

注意，其实我们在上文中显示了一个覆盖率准则，即每个类，无论是有效的或无效的，都至少要在一个测试用例中测试到。

在等价类划分中我们使用了什么缺陷假设？在大多数时候，我们在寻找系统无法适当处理某些等价类的情况。比如说当系统应该拒绝某个输入值的时候却接受了这个输入值，或者在应该接受某个输入值的时候却拒绝了这个输入值，又或者当系统正确地接受或拒绝了某个值，但却误用对另一个等价类的处理方式处理了该值。

ISTQB 术语

等价类划分：根据规格说明，输入域或输出域的一个子集内的任何值都能使组件或系统产生相同的响应结果。

等价类划分技术：黑盒测试用例设计技术，该技术从组件的等价类中选取典型的代表值进行测试。原则上每个等价类中至少要选取一个典型的代表值来设计测试用例。

您可能会觉得从文字上理解这个概念有点模糊不清，那我们试试从图形来理解。图 4-2 展示了如何使用图形来解释等价类划分。

在图 4-2 的上半部分，我们对一个集合进行了划分。这个集合可以是一个输入域、一个输出域、一个测试前置条件或后置条件、一种配置或其他任何我们感兴趣的东西。关键是我们使用等价类划分技术将一个集合划分为两个不相交的子集，子集中的每个个

pin51.com 最新最全 拼吾爱

体都共享一些显著的不同于其他子集的特性。例如，一个简单的画图系统，可以将图形染成红色、绿色或蓝色，因此您可以将“染色”集合划分为 3 个不相交的子集：红色、绿色和蓝色。

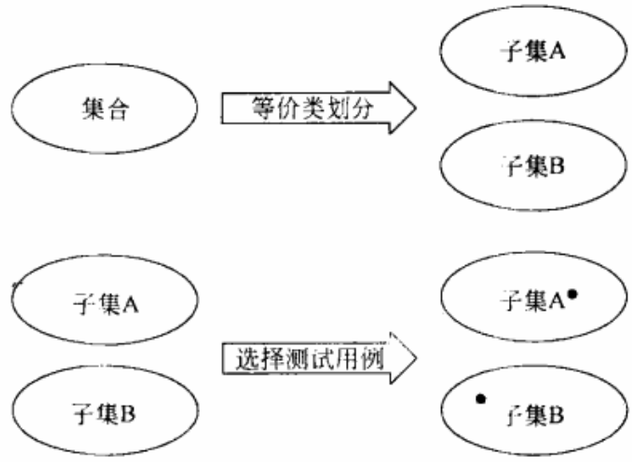


图 4-2　等价类划分技术的图形描述

在图 4-2 的下半部分中，我们从子集中选取了测试用例的值。小黑点代表了我们从各子集中选取的代表值。这意味着至少要从每个子集中选取一个代表值。纯粹的等价类划分技术并未规定选取什么值作为代表值，因此您可以从子集中选取任何您喜欢的值作为代表值。不过您有可能会想“也许选某些值的效果比其他值更好”，这样是允许的。对于这一点，您可以留一点悬念在这里，稍后我们会回到这个话题上来。

在此我们可以开始设计其余的测试用例。如果我们划分的集合是一个输入域，那么我们应该参考需求规格说明，了解系统如何处理每个子集。如果我们划分的集合是一个输出域，那么我们应该参考相关需求寻找触发那个输出值的输入值。我们也可以使用其他测试技术去设计剩余的测试用例。

图 4-3 中，我们迭代地使用了等价类划分技术。在图中，我们对其中一个子集进行

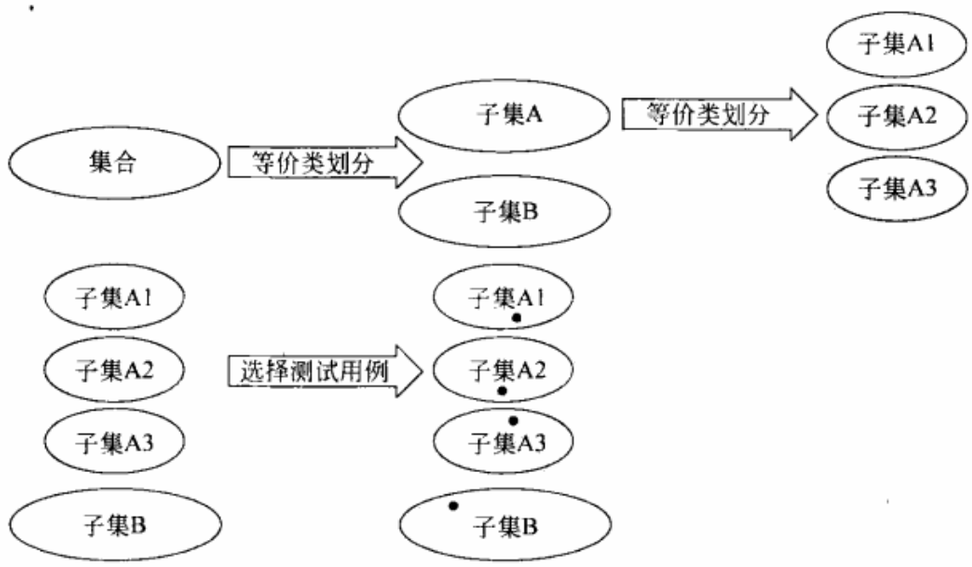


图 4-3　子划分

了第二轮划分，将它划分成了 3 个更小的子集。此时，我们的测试用例选取 4 个代表值，子集 B、A1、A2、A3 各一个代表值。注意我们不用再从子集 A 中选取代表值了，因为子集 A1、A2、A3 中的任何值同时也是子集 A 的值。

4.2.2 避免错误的等价类划分

虽然等价类划分技术比较简单，但是人们在使用的时候总是会犯一些常见的错误。接下来我们讲讲这些常见的错误，以便您能避免这些问题。

第一点，如图 4-4 中的上半部分，子集必须是不相交的。即任意两个子集不能有相同的值。等价类划分技术的关键是测试一个系统是否能以不同的（当然也应该是恰当的）方式处理不同的情况。如果无法确定应该使用哪种处理方式，那么我们应该如何设计测试用例？“试一下，然后看看发生了什么？”这不能算作测试的！

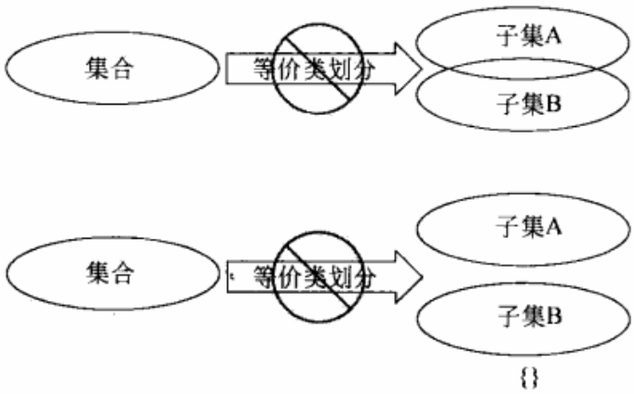


图 4-4 等价类划分过程中的错误

第二点，如图 4-4 的下半部分，任何子集不能为空。也就是说，如果划分出了一个空子集，那么这种划分是没有意义的。因为我们无法从一个空子集中选取代表值。

第三点，没有在图中画出，因为我不知道怎么用图去清楚地解释这一点。记住，等价类划分技术是在进行划分而不是在做减法。我的意思是，根据数学集合理论，等价类划分技术生成的子集的合集应该等同于原始集合。也就是说，在等价类划分过程中我们不会丢弃那些多余的子集——至少，在我们正确地进行等价类划分的时候不应该发生这种情况。

记住这点很重要，因为一旦我们犯了这个错误，那么我们很有可能会在测试设计过程中漏掉那些输入、输出、配置或其他项的重要的子集。

4.2.3 用等价类划分技术生成测试用例

如图 4-5 所示，当我们对多个集合进行等价类划分的时候，我们会从这些集合中各选一个子集组成测试用例。这里我们用两个集合进行说明，X 集合和 Y 集合。我们将 X 集合划分成两个子集，X1 和 X2，将 Y 集合划分成 3 个子集，Y1、Y2 和 Y3。我们从 X1、X2、Y1、Y2、Y3 这 5 个子集中选取测试用例所需的值。由于我们可以将 X 的子集和 Y 的子集的值组合在一起，这样的话，仅 3 个测试用例就可以满足等价类划分技术的

pin51.com 最新最全源代码 拼吾爱

基本要求（假设这些值都是独立的，并且有效）。

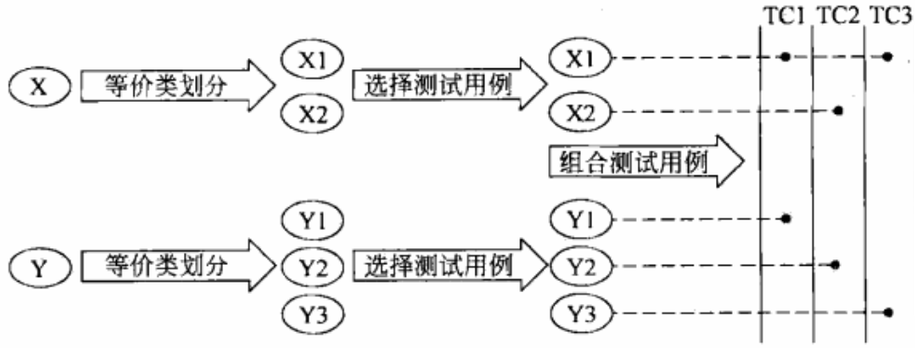


图 4-5　用有效值设计测试用例

例如，想象一下您正在测试一个基于浏览器的应用程序。您希望测试两种浏览器：Internet Explorer 和 Firefox，以及 3 种网络连接类型：拨号、DSL 和调制解调器。由于浏览器和连接类型是独立的，因此我们可以设计 3 个测试用例。每个测试用例中包含了一个连接类型和一个浏览器类型。其中有一种浏览器类型会在两个测试用例中出现。

在描述图 4-4 的时候，我提到当子集相互独立且为有效值的时候，我们可以从这些子集中选择代表值合并组成测试用例。当然，这是种理想状态，在现实情况中并非尽然。

在某些情况中，子集之间的值并不是独立的，对一个子集中值的选择限制了对另一个子集的值的选择。例如，您在测试应用程序和操作系统的组合。如果该应用程序没有基于某个操作系统的版本，那么您就不能在该操作系统上测试那个应用程序。

在某些情况中，子集包含的值是无效的。例如，在图 4-5 中，假如我们在测试一个项目管理应用程序，比如 Microsoft Project。假设 X 是我们正在处理的事件类型，可以是一个任务（X1）或一个里程碑（X2）。假设 Y 是这个事件的起始日期，可以是过去（Y1），现在（Y2）或将来（Y3）。假设 Z 是这个事件的结束日期，可以是起始日期的当天或之后（Z1），也可以是在起始日期之前（Z2）。当然，Z2 是无效的，因为没有哪个事件的持续时间可以是负值。

我们在 TC1、TC2 和 TC3 测试用例中测试了任务和里程碑与过去、现在、将来以及有效的结束日期的组合情况，如图 4-6 所示。在 TC4 中，我们测试系统是否能拒绝无效的结束日期。我们输入了一项任务，该任务的起始日期为过去，但它的结束日期在起始日期之前。如果我们想对无效状况进行再划分，那么我们还应增加两个测试用例。一个测试用例的起始日期是现在，结束日期在起始日期之前。另一个测试用例的起始日期在将来，结束日期在起始日期之前。

在上述的特定例子中，只有一个集合中的一个子集是无效的。然而在大多数情况中，很多甚至所有集合都有无效子集。比如在测试一个电子商务应用程序的时候，一个典型的结账界面会有很多域，对于这些域，通常至少有一种方法能让这个域失效。当有多个无效值的时候，首先我们应该在每个测试用例中设计选择单个无效值。这样，我们才能测试系统是否能正确地处理每一个无效值，比如系统正确地拒绝任何指定的无效输入值或进行某种相应的异常处理。

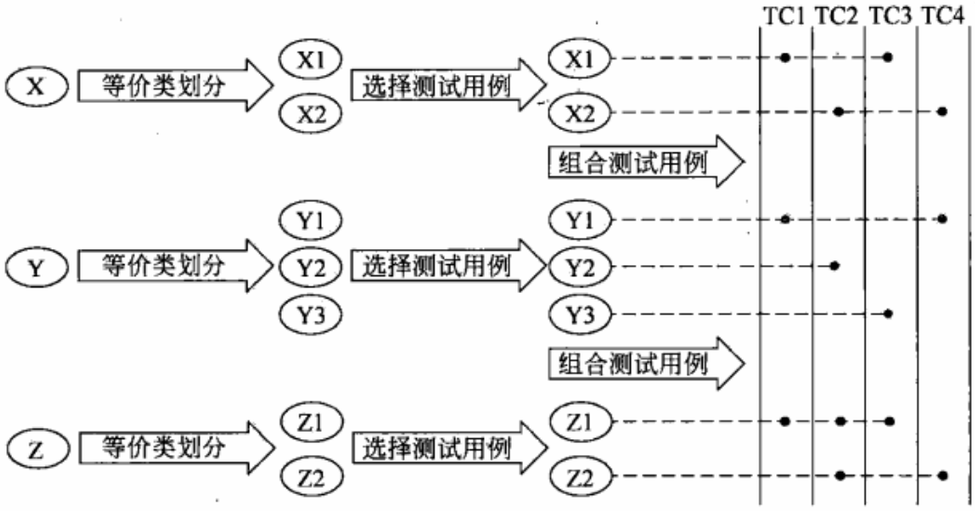


图 4-6 用无效值设计测试用例

我说过首先要在每个测试用例中只设计测试一个无效值。当然，在单独地测试过每个无效值之后，如果想测试各种无效值组合的情况，您也可以这么做，只要有合理的理由，比如覆盖某种风险。不过说实话，您这么做大多数时候只是在花很多测试时间去测试一些不是很重要的东西而已。

下面是对系统配置中的单一值进行等价类划分的一个例子。在我先前提到的网络应用项目中，该应用程序可以有 4 种配置。给小孩用的、给青少年用的、给成年人用的或给老人用的。这个配置的值存储在因特网服务提供商的服务器的数据库中，当一个该应用程序接入因特网的时候，该应用程序就可通过选择的配置值获取相应的配置。

根据配置的值，系统的预期行为有两个关键的不同点。第一点，对于儿童和青少年，系统会有一个过滤功能。这个功能将决定系统能访问哪些网站以及不能访问哪些网站。对于儿童，系统将使用最严格的设置，而对于青少年，使用较严格的设置。当然，对于成年人和老年人，系统就不会进行任何过滤，他们能浏览任何网站。

另一点是，这 4 种配置各自有一个包含默认电子商务网站列表的所谓网上商城。这些默认的电子商务网站是由专业市场团队按照适合的年龄段选择出来的。

以上这些是我们预期（针对不同年龄段配置）的区别。除此之外，我们也意识到某些缺陷类型的固有本性会导致奇怪的、意料外的事情发生。例如，在该系统中，不同配置的性能应该是一样的，但是由于在小孩和青少年的系统中使用了过滤器，因此他们在使用系统的时候可能会感到侵入式过滤功能带来的系统响应慢的问题。对于这些不当的系统行为我们也应予以关注。

因此，为了测试各种配置之间预期之外的差别，我们要保证在测试全过程中，在实验室至少部署每种配置一次，并且在该配置下或多或少随机地执行不受配置影响的测试。（在本节的后续部分我们会讲解如何测试配置的组合。）而对于测试那些可预期的差别，例如过滤功能和电子商务网站列表，我们只需确定我们正在正确的配置上进行相应测试即可。但我们还是面临一些挑战，例如，当测试网上商城时的预期结果是确定时，在这里就是市场团队提供的每种配置的具体商务网站列表，但过滤功能的预期结果不是确定的，而仅是一个逻辑值。这可能导致测试执行时产生大量恼人的缺陷报告，比如我们曾

pin51.com 拼吾爱 最新最全源分享

经发现很多能够创造性地绕开过滤器，访问年龄不当网站的方法。

4.2.4 等价类划分技术练习

图 4-7 是 HELLOCARMS 系统中某一个界面的原型。在这个界面中要求输入三方面的信息：

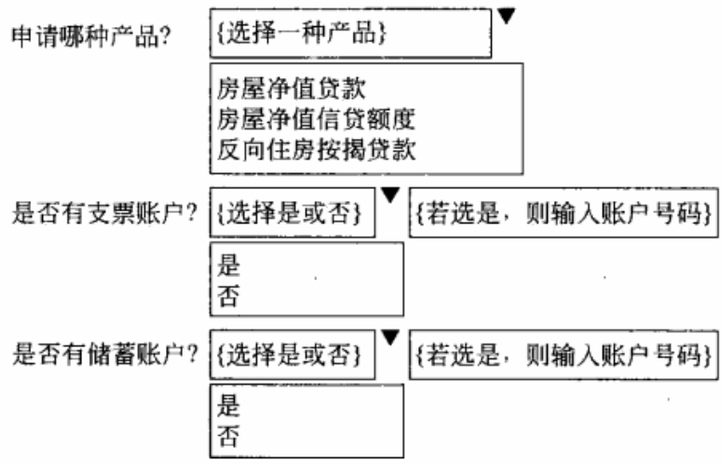


图 4-7 HELLOCARMS 系统的界面原型

- 产品的应用领域：
 - 房屋净值贷款。
 - 房屋净值信贷额度。
 - 反向住房按揭贷款。
- 是否有一个全球通用的支票账户：是或否。
- 是否有一个全球通用的储蓄账户：是或否。

如果用户选择有全球通用的账户，那么他必须输入相应的账号。然后系统会通过银行中央数据库验证用户输入的账号。如果用户表示没有这样的账户，那么他必须将这个域置为空。

如果这些域中的输入有效时，包括账号输入域，那么系统接受这些输入。只要其中有一个域无效，系统就会显示一条错误信息。

本练习包含两个部分：

1. 对上述三方面的信息进行等价类划分，包括有效的和无效的类。
2. 用我们刚提到的有效值和无效值组合的方法设计测试用例覆盖这些划分。

上述两个部分的答案就在下面，不过在您检查自己第二部分的答案之前，您应该好好检查一下自己第一部分的答案，以和您自己的答案对照（如果有必要的话，修改自己的答案）。

4.2.5 等价类划分技术练习题答案

首先，我们看看如何进行等价类划分。

对于应用领域这个选项，等价类划分如下：

pin51.com 新客发 新客服务源码下载

#	划　　分
1	房屋净值贷款
2	房屋净值信贷额度
3	反向住房按揭贷款

注意在上述的屏幕原型中这些信息是以选项列表的方式表示的，因此用户无法在此输入一个无效的产品。

对于其他两个账户域，理想情况下最好为一个包括两个子域的输入域。第一个子域是“是/否”域，这个子域决定了第二个子域账号输入域的验证规则。如果第一个子域的值是“是”，那么第二个子域必须是一个合法的账号。如果第一个子域的值是“否”，那么第二个子域必须为空。

因此，支票账户信息的划分如下：

#	划　　分
1	是-有效
2	是-无效
3	否-置空
4	否-置为非空

储蓄账户信息的划分：

#	划　　分
1	是-有效
2	是-无效
3	否-置空
4	否-置为非空

注意，在上述两个划分中，第 2 项和第 4 项都是无效的划分项，而第 1 项和第 3 项是有效的划分项。

现在，让我们为这些等价类划分创建测试用例。在这个过程中，我会记录相关信息以保证测试用例号与等价类划分之间的可追溯性。如果每一个等价类都有一个对应的测试用例，这表明我已经仔细地将有效划分和无效划分组合在了一起。

输　入	1	2	3	4	5	6	7
产品	HEL	LOC	RM	HEL	LOC	RM	HEL
存在支票账户？	是	否	否	是	否	否	否
验证账户	有效	置空	置空	无效	置为非空	置空	置空
存在储蓄账户？	否	是	否	否	否	是	否
验证账户	置空	有效	置空	置空	置空	无效	置为非空
输　出							
接受？	是	是	是	否	否	否	否
错误？	否	否	否	是	是	是	是

pin51.com 最新最全源代码 拼吾爱

产品：

#	划　分	测试用例
1	房屋净值贷款	1
2	房屋净值信贷额度	2
3	反向住房按揭贷款	3

支票账户：

#	划　分	测试用例
1	是-有效	1
2	是-无效	4
3	否-置空	2
4	否-置为非空	5

储蓄账户：

#	划　分	测试用例
1	是-有效	2
2	是-无效	6
3	否-置空	1
4	否-置为非空	7

您应该注意到上述测试用例并没有覆盖到所有的输入组合。例如，我们并未测试当一个人既有合法的储蓄账户又有合法的支票账户的情况。然而这个测试可能是有意义的，因为不同时期创建的账户的信息可能存在冲突，比如在某些国家妇女在结婚后通常会改成丈夫的姓。在上述测试中，我们也没有测试无效账户组合在一起的情况，或者两个账户都有效，但组合在一起时无效的情况，例如两个属于完全不同人员的账户。

我们将在本节的后续部分讲解测试条件组合的技术。

4.2.6 边界值分析

接下来，让我们利用边界值分析技术来改善上述等价类划分技术。就概念而言，边界值分析就是测试等价类的边界。也就是说，我们不再随意地从等价类中选择一个值作为代表值，而是测试其中的最大值和最小值。

边界值分析使用的模型可以是图形模型或者是数学模型，不过其原理都是要识别两个等价类之间的两个边界值。（在某些技术中可以识别出3个边界值，我会在后面提到）现在，我们先不讨论等价类之间是否存在边界的问题，我会在后面提到这点。我们先假设确实存在边界值，这个边界值通常是等价类的一个特殊元素，在这个边界值的下一个值就是其他的等价类或者系统的预期行为会有所改变。如果边界值是一个有效等价类的元素，那么显然它们是有效的，如果是一个无效等价类的元素，那么它们是无效的。

从等价类中选择边界值来设计测试用例的方法与不考虑边界值时的方法非常类似。

我们将有效的边界值组合成测试用例，然后再用一个无效的边界值与其他有效的边界值组合。

与先前等价类划分技术相似，所有边界值都应该在测试用例中提到。也就是说，覆盖率标准是每个边界值，无论是有效的和无效的，应至少在一个测试用例中被测试到。

不同的是，由于每个等价类至少有两个边界值，因此我们会有更多的测试用例，大约是先前的两倍。

ISTQB 术语

边界值分析：一种黑盒设计技术（black box test design technique），基于边界值进行测试用例的设计。

边界值：通过分析输入或输出变量的边界或等价划分（equivalence partition）的边界来设计测试用例，例如，取变量的最大值、最小值、中间值、比最大值大的值、比最小值小的值等。

既然会产生更多测试用例，除非有合理的理由，没人会喜欢增加工作量和成本。那么，我们为何要设计这些额外的测试用例呢？这与边界值分析所使用的缺陷假设有关。由于我们在测试等价类划分的元素——每一个边界值都是一个等价类的元素——我们在测试系统是否无法正确处理某些等价类。在此，无法正确处理的情况是指接受了无效值，或拒绝了有效值，或正确处理了输入，但后续的处理出了问题，比如输入值是属于某个等价类，而系统将其当成另一个等价类处理了。然而，通过测试边界值，我们就能确定是否正确定义了等价类之间的边界。

所有的等价类都有边界值么？当然不是。边界值分析技术是等价类划分技术的拓展，只有当等价类的元素是有序的情况下我们才会使用这种技术。

那么，有序是指什么呢？嗯，一个有序集是指当元素不相等的情况下，某个元素大于或小于其他的一些元素。这种有序必须是有意义的，比如在下拉菜单的代码中，某个项在其他项的上方或下方，并不意味着这些项之间有大于或小于的关系。

4.2.7 等价类划分技术和边界值分析的样例

让我们举一些例子来看看在划分等价类的时候什么时候可以使用边界值分析，什么时候不可以使用边界值分析。

在图 4-8 中您可以看到一个 Insert 下拉菜单，这个菜单位于 PowerPoint 软件界面顶部的菜单栏上。这是一个典型的下拉菜单。

在这种情况中，每个选项自身就是一个等价类。虽然在这个菜单上有第一个选项和最后一个选项，但这些选项的顺序是随意的，因此在逻辑上并没有次序之分。在软件的内部，菜单上的这些项，也不太可能有“大于/小于”的关系。所以在这里我们只需等价类划分技术即可（我们必须测试菜单上的每个选项），无需再进行边界值分析了。

注意，这里显然不可能有“无效”选项，因为所有我们不能选择



图 4-8 下拉菜单

pin51.com 最新拼吾爱

的选项都已经变成灰色（失效了）。这个能起作用么？不一定。因此我们在测试这些选项的时候就应确保那些应该变成灰色的选项确实变成了灰色，那些不应该变成灰色的选项没有改变颜色，并且那些变成灰色的选项应该是不能访问的。

因此，如图 4-9 所示，对这个菜单我们将把它的每个选项作为一个等价类。注意，此处有一些选项是无效的。

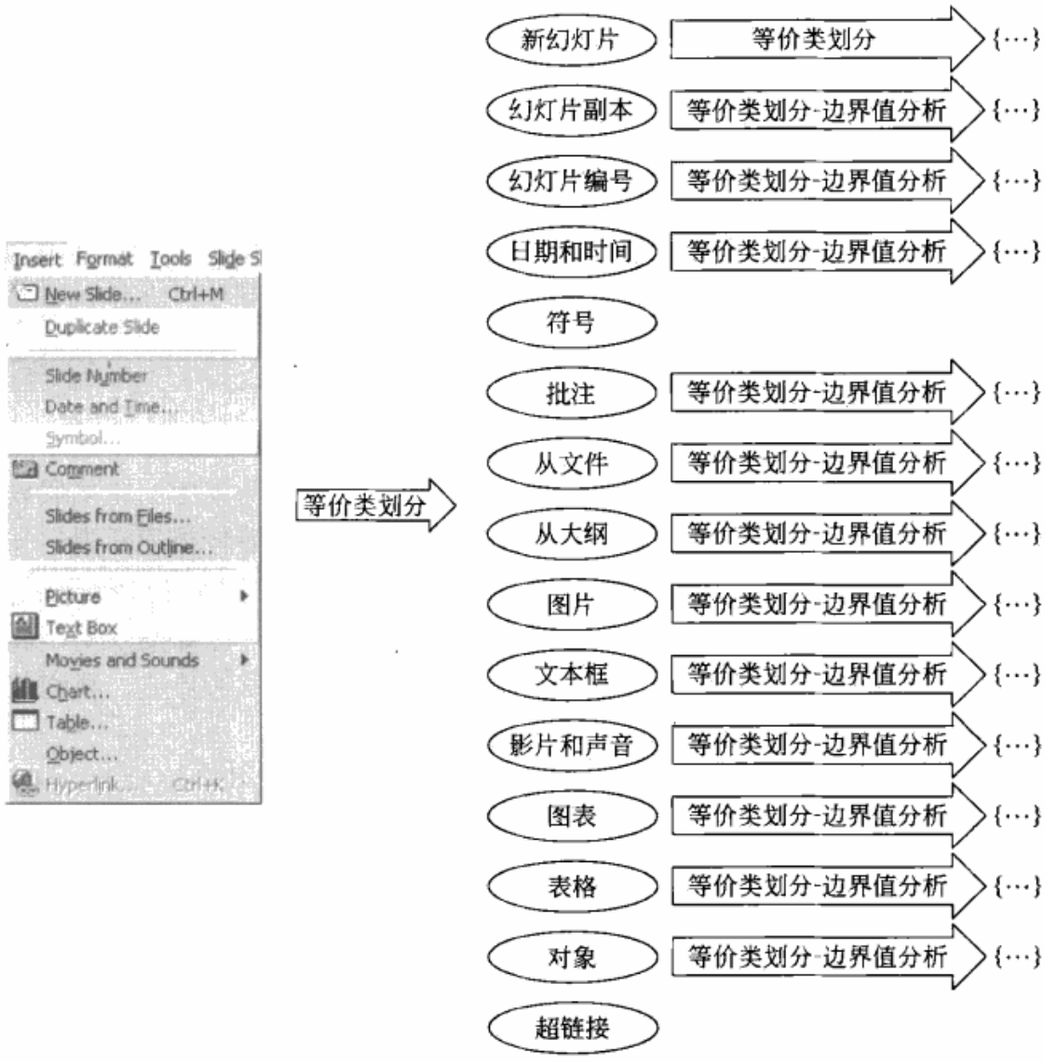


图 4-9　下拉菜单示例

对于每一个有效的菜单选项，我们都要进行进一步的等价类划分，大多数情况下，还要进行边界值分析。例如，我们可能要测试插入来自文件的幻灯片（Slides from Files）选项，那么就应测试只有一个幻灯片的情况，以及最大数目的幻灯片的情况（无论这个数目是多大）。同时我们还要测试文件中有零个幻灯片的情况（如果可能的话）以及文件中包含最大数目加一个幻灯片的情况。

并不是所有的下拉菜单都是无序的集合。有些菜单使用了整数域。整数是有序集合。例如，Microsoft Word 软件的字体大小，如图 4-10 所示，您可以直接输入数值，也可以通过下拉菜单选择。

对输入字体大小的方式进行等价类划分可以得到：

pin51.com 拼吾爱 资源分享

- 选择菜单选项。
- 在域中输入值。

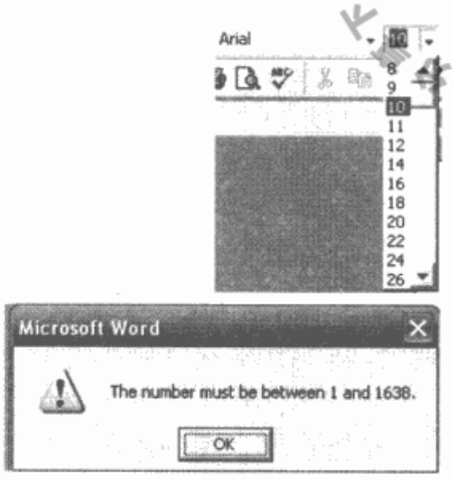


图 4-10　整数域

您可能会说，“嘿，这些集合是相交的，因为我可以先选择，然后再输入，或先输入，然后再选择。”的确是这样，不过在任何时候您只能选择这两种方式中的一种（如果您是在设置字体大小）。对于您所说的那种情况，我们在后面会介绍一种状态转换测试技术。

现在，我们可以对这两种字体大小的输入方式进行边界值分析。在 Word 中，字体大小是整数值，而整数值是有序集合。因此我们可以选择最大字体和最小字体的菜单选项，也可以直接输入最大字体或最小字体。

字体大小的最大值和最小值是什么？理想情况中，我们可以在需求规格说明中找到答案。除此之外，我们也可以做一些试验，如图 4-10 所示，最小的字体大小是 1，最大值是 1638。的确是这样么？谁知道呢？我所了解的是 Arial 这种字体的最大值和最小值。字体的种类可能会影响字体大小的最大值和最小值。难道我必须测试每种字体的极限值？这绝对将是测试人员的噩梦！因此我们最好在测试之前就弄清楚这些值到底应该是什么。

最后，在这个菜单中您是无法选择无效值的，不过您可以输入无效值。还有可能存在的一个问题是下拉菜单中没有给出最大值和最小值的选项。从可测性的角度而言，让用户选择一个菜单项比允许他们直接输入数据或修改域要好得多。

在图 4-11 中，我对字体大小这个域进行了等价类分析和边界值分析，直到生成测试输入数据。

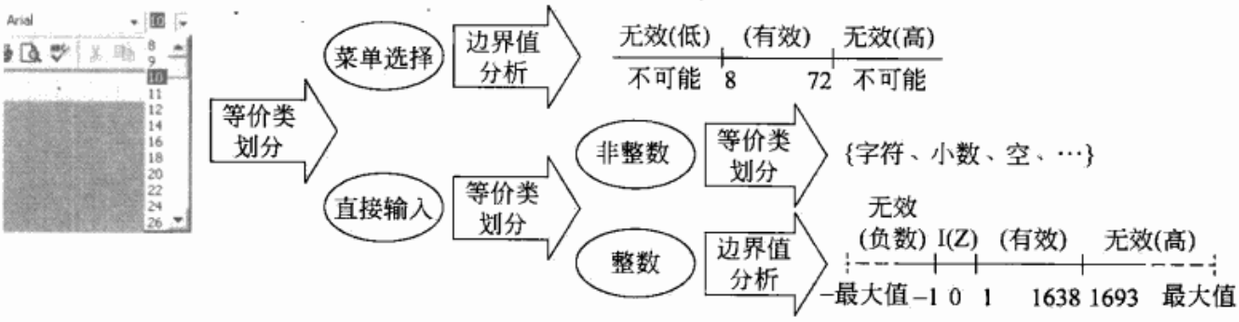


图 4-11　整数示例

生成的测试输入数据如下：

- 如果选择菜单选项，那么我们应该测试 8 和 72：任何更小或更大的值都是不可能的，至少菜单上没有这个选项。
- 如果直接在域中输入值，那么我们可以先考虑非整数的无效等价类划分。我们可以输入字母 *a*、数字 6.5、空值（比如，将域置为空）、斜杠、减号和加号。我们还可以输入几乎无限长度的无效值，不过这好像有点傻。
- 对于整数值的等价类划分，我们可以输入–1、0、1、1638（错误消息中给出的最

pin51.com 最新最全 拼吾爱

大有效值），1639（大于最大值的最小的无效值），一些大的正数和负数。那么此处大的数字是指什么呢？一般而言，我们会使用记事本或其他简单的文本编辑器创建 10 个 9 组成的字符串，然后再拷贝到相应的域中。如果系统接受了这个输入，那么再尝试 20 个 9 组成的字符串，再 40 个 9，100 个 9。这个也好像有点傻。

问题的答案在于“对于这些无效输入我应该测试到什么程度？”，当然，这就需要考虑风险问题。利用无效输入触发缓存溢出错误是一种常见的安全攻击方法。如果您的应用程序要处理安全相关问题或隐私数据，例如客户账户信息，那么对您系统的每一个域执行一系列精心设计的缓冲区溢出测试是有意义的。然而，对于 Word 这种程序而言，花费大量时间在字体大小这个输入域中测试各种不同类型的无效字符串是没有什么意义的。聪明点吧！在确定测试程度的时候要考虑风险因素。

回忆一下，在 IEEE 829 测试用例规格说明模板中要求测试用例要包含输入和输出。因此，我们预期系统会拒绝无效输入并给出一个有用的出错信息。对于有效的输入，我们期望系统会修改被选中的文本的字体大小。

整数估计是测试人员遇到的最简单的有序集合了。我们有时候还得测试浮点数。浮点数也叫做实数或十进制小数。浮点数和整数一样，也是有序集合，区别在于浮点数有小数位，而整数没有。

这随即引出一个重要问题，有多少小数位？当我们在处理某些域的精度时就会遇到这个问题。这有时候也被称为 ε（epsilon）或最小可辨差别。

之所以要重视精度问题是基于两个理由。第一，在不知道精度的情况下无法明确边界在哪里。第二，当精度不明确尤其是模棱两可的时候，很容易滋生大量缺陷。

大多数软件，根据其使用的编程语言的特性，会有一个内建的浮点数据类型。问题是该浮点类型的精度在不同的计算机上是不同的。另外一个潜在的问题是，浮点数据类型的范围，比如其最小值和最大值，是可以变化的。

在不深入技术细节的情况下，浮点数还存在另一个问题，即它的存储方式。精度并不是像您想象的那样通过小数点前后的数字给出的。精度是指所有的有效数字。在有效数字和小数点之间可能会有大量的零，例如，999 999 兆兆是一个非常大的数字，但可能只有最前面的 6 位，即 999 999，会被作为有效数字存储。这意味着，对于那些非常大的正数和负数和非常小的正数和负数，例如非常接近零的数，在取整数的时候，我们会遇到一些很有趣的错误，尤其是在计算的时候。

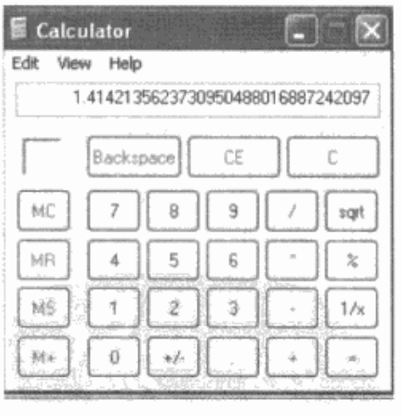


图 4-12 浮点数

我在 UCLA（加利福尼亚大学洛杉矶分校）攻读计算机科学学位的时候花了整个季度学习数值分析。其中大部分时间就花在处理浮点数计算问题上了，粗心大意的程序员在这个地方会犯很多错误。

图 4-12 是微软的计算器小程序。默认的，主要的输入域是一个浮点数域。对于浮点数，微软提供了一种叫

做拓展精度的功能，而不是依靠浮点数的内部表示。如图所示，这种功能为我们提供了至少 32 位的精度。另外，浮点数，即那些能写成分数形式的数，比如 1/2 或 3/8，依旧是以分数的形式存储的，进一步提供了计算的准确性。当然，有无限小数位的无理数比如 π 和其他常量，由于会被截成固定的位数，因此在任何系统中都是不准确的。

在图 4-13 中，您可以看到对计算器输入域进行等价类划分和边界值分析以测试其平方根功能，直到生成测试输出的过程。平方根接受任何大于等于零的浮点数输入，并返回一个数，这个输出值的平方应该就是原来的输入值。

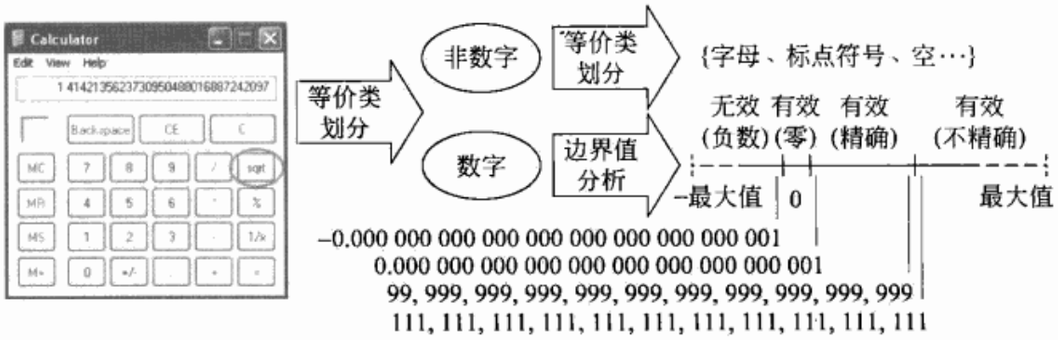


图 4-13　浮点数示例

通过应用等价类划分技术和边界值分析，我们得到了下列测试输入值。

就像先前测试整数时一样，我们可以测试一些非数字输入比如 *a*，ε，空（如果可能的话）等等。对于这个非数字的等价类您可以进行无限地划分，因此记得使用风险分析。

对于数字的浮点数输入，我们有 4 个等价类，其中 3 个是有效的等价类：

1. 零自身就是这个等价类的上边界和下边界。

2. 小于等于 32 位的数字，因此应该能精确地表示。我们只用一个非常接近于零但不等于零的数，以及一个有 30 个 9 的数，作为边界值。

3. 大于等于 33 位的数字，因此无法精确地表示。创建这种数字可能要花一些时间，因为您先得输入一个最大的数字，然后乘以其他数字，最后得到更大的数字。我花了几分钟去测试这一项，不过并没有测试出计算器什么时候会溢出。我发现当计算器用于计算的时间过长的时候，它会询问是否要继续计算。最好能在需求规格说明的某个地方说明这一点。

有一个无效类，负数。包括非常接近于零的负数和最大的负数，当然，在没有需求规格说明的情况下要找出这个最大负数可能要花一点时间的。

这是一个金融应用程序 Quicken 中一个投资账户的屏幕截图（见图 4-14）。测试金融操作的时候包含了浮点数的计算，因此在这里可以应用我们讨论过的方法。然而要注意的是，精度问题会导致很多严重的实际问题。在涉及钱、投资账户平衡表以及其他类似问题的时候，人们对于取整错误会显得非常敏感。此外，在履行一些规定和合同时也会对这个问题有特定的要求。因此在测试金融应用程序的时候对于类似的问题要非常小心。

让我们仔细瞧瞧。在图 4-15 中，我们对 Quicken 程序的股票交易的输入进行了等价类划分和边界值分析。在 Quicken 程序中，若要进行股票交易，先要输入股票代码，然后是股票的数目，每股价格以及手续费（若有的话）。Quicken 程序会计算出整个交易的费用。

pin51.com 最新最全源代码 拼吾爱

价格	股票	账户		Clr	股票盈亏	
		佣金/费用				
9.81	4.778	46	87	R	1 197	35
9.77	5.072	49	55	R	1 202	42
9.74	5.331	51	92	R	1 207	75
9.74	5.331	51	92	R	1 213	08
9.82	4.652	45	68	R	1 217	74
市值：	18 958.56	最终股票盈亏：			2 004.076	

图 4-14　Quicken 投资账户截图

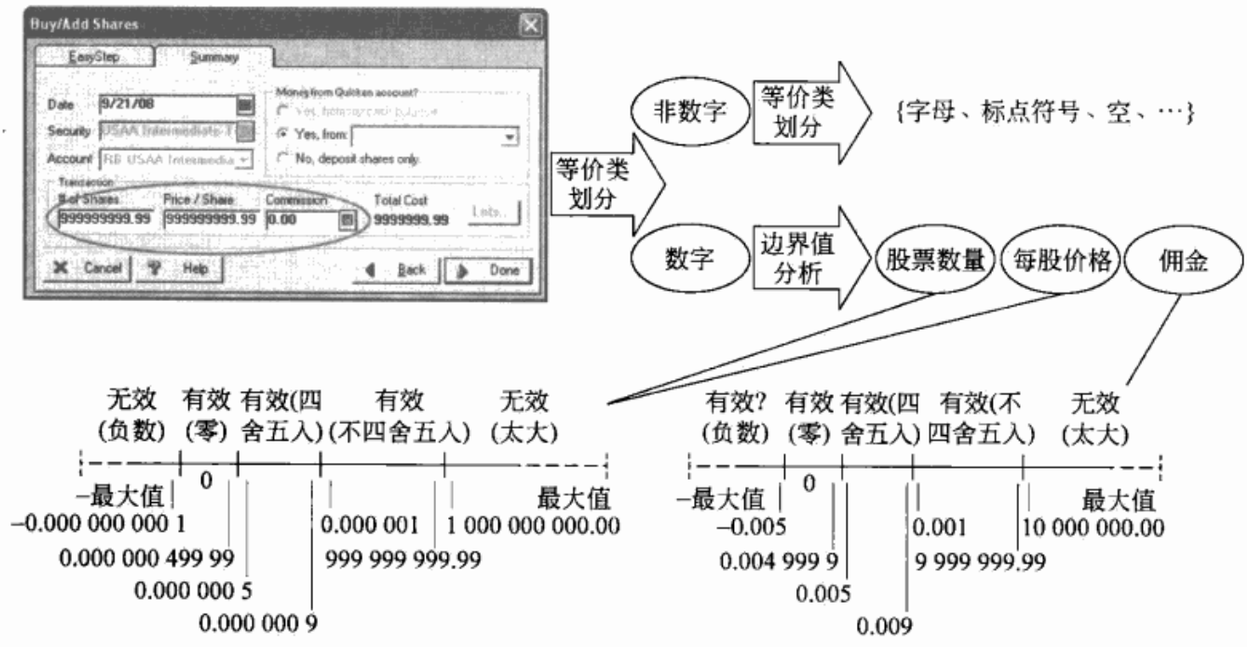


图 4-15　Quicken 股票交易

首先，对于股票数、每股价格和手续费这 3 个域，任何非数字输入都是无效的。

对于数字输入，系统将会以相同的方式处理每股价格和股票数这两个域。这个数字必须大于等于零。注意，有时候虽然我们输入了非零数字，但是系统会将这个数字取整为零。对于大于零的数，我们最小要输入 0.000 000 5，系统会将其取整为 0.000 001。

对于手续费，也是类似的，为便于（用户）理解，系统会取整到美分。在图 4-15 中我不大理解的是手续费居然可以是负数。我估计在某些情况下这是可能的，不过我从来没这么幸运过！

在图 4-15 中有很多信息，因此您可能要花一些时间去学习我所识别出的测试输入值。

您也可以对总成本这个域进行同样的分析。不过，这是一个输出域，因此您只能通过创建适当的输入以生成您要的输出。从图 4-15 中您可以看到显然很难做到这一点。

也许您会使用这些测试输入值设计一些测试用例，用于测试股票交易。我们会在讨

论用例的时候再进一步地谈谈为什么要设计这些测试用例。

日期输入域（如图 4-16 所示）和时间输入域是有序集，因此等价类划分技术和边界值分析技术我们都可以使用。但是它们比整数和浮点数更复杂，因为日期和时间输入域有两个或超过两个的子域。比如日期域，它的一个或两个子域决定了另外一个子域的有效范围。又比如时间域，域的格式决定了其中一个子域的有效性。

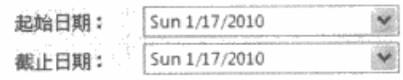


图 4-16 日期和时间

接下来的消息对测试分析师而言喜忧参半。绝大部分的应用程序使用标准库来验证日期和时间输入，因此我们不必进行大量测试。然而，应用程序经常接受两个或多个日期或时间输入，这些输入之间会有一些内在的关系，比如飞机航班的到达日期和起飞日期。因此大多数测试分析师会将精力放在测试日期之间的关系上，这可能会比较复杂。

在图 4-17 中，可以看到我首先对 Outlook 程序的任务对话框的起始日期域和预期结束日期域进行了边界值分析然后进行等价类划分。接下来我们来看看如何设计这些测试的输入值及其含义。

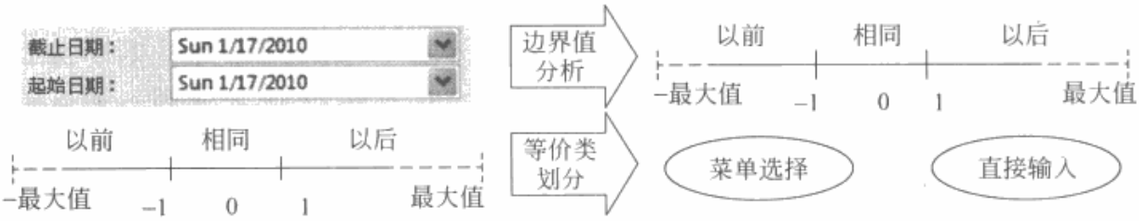


图 4-17 日期和时间示例

首先，通过边界值分析生成的测试输入值使得我们能测试预期结束日期和起始日期之间的关系。比如预期结束日期和起始日期在同一天，这种情况下的输入是有效的。或者预期结束日期比起始日期晚，这也是有效的。具体晚多少时间可以是一天或某一个最大值。我们应该能在需求规格说明文档上找到这个最大值。

对于无效输入，我们可以试着输入一个比起始日期早的预期结束日期。这绝对是无效的。可以是早一天，也可以早某一个最大值。

现在，我们可以对日期的输入方式进行等价类划分。由于日期输入方式对验证过程有影响，因此我们应该测试不同的日期输入方式。下拉日期菜单并从中选择日期会立刻出发验证功能，而手动输入的日期要到保存任务的时候才进行验证。

大多数情况下人们是在自由格式的、可编辑的域中输入不同长度和内容的字符串。验证这些域的规则取决于这是个什么域。边界值和等价类通常都不是来自于数据自身的性质，比如我们刚才的整数和浮点数等，而是来自于应用程序为这些数据设置的规则。因此，在某些情况下字符串会有边界值，特别是就长度而言的时候，而在另一些情况下就没有边界值。

对于日期而言，也有可能存在这样的情况：当一些域单独处理的时候是有效的，而放在一起的时候就无效了。图 4-18 是 RBCS 网站的支付处理界面。上面的 6 个域的信息

pin51.com 最新最全源码 拼吾爱

都是正确的，但它们并不属于同一个实际的信用卡。图 4-19 中是对信用卡类型和卡号进行等价类划分和边界值分析。

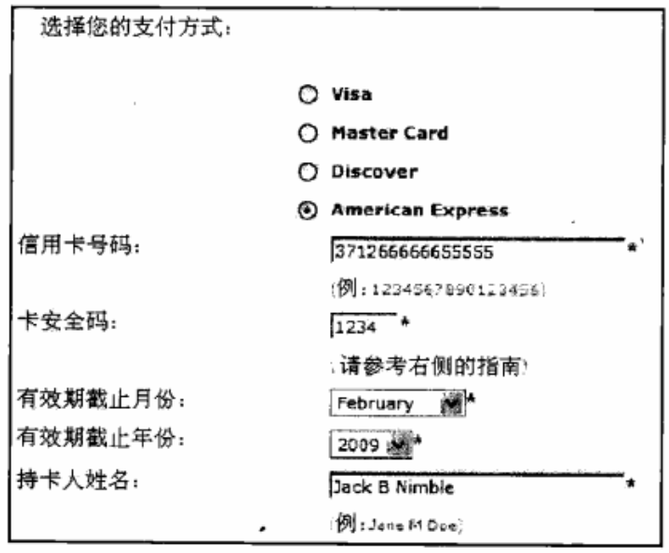


图 4-18　RBCS 网站的电子商务支付处理界面

从图 4-19 中可以看到，每一种信用卡类型就是一个等价类。之所以必须这么做，是因为每种类型的信用卡它们初始的几位数字以及卡号的位数都是不同的。对于每一种信用卡类型，我们要尝试一下一些常见的非数字输入。对于数字输入，要进行边界值分析以确定其有效性。共有 4 个等价类：

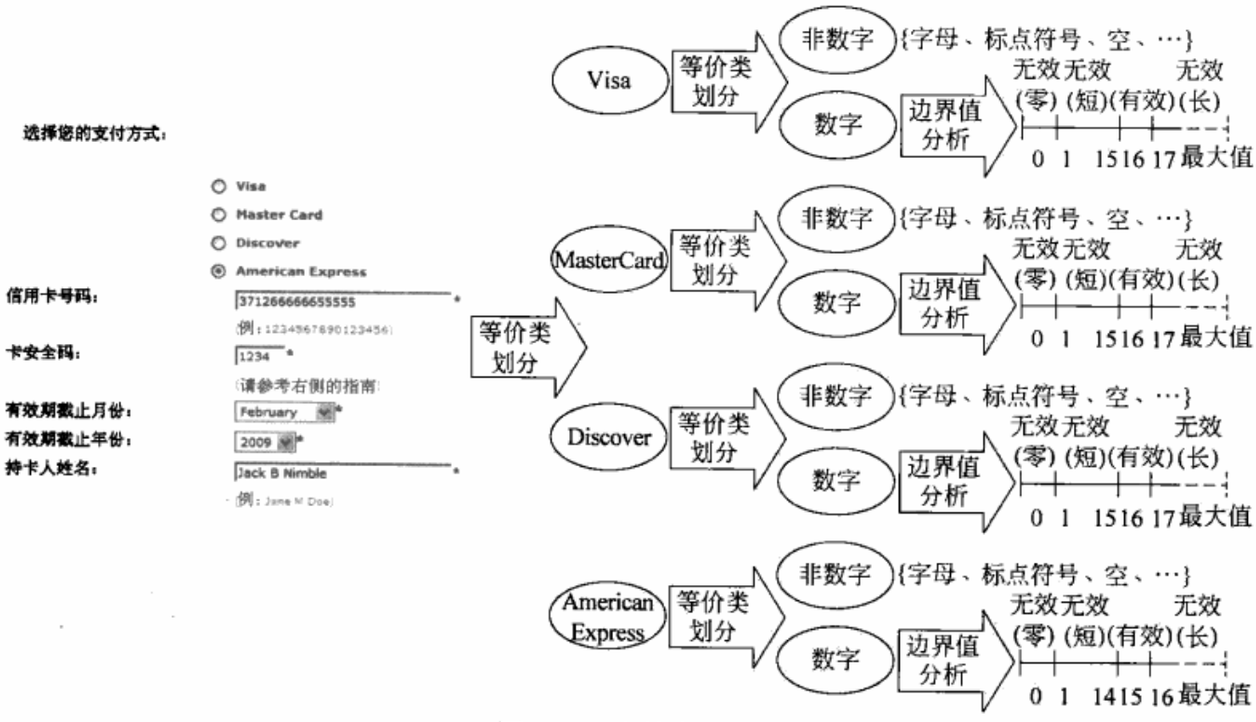


图 4-19　字符串示例

1. 卡号长度为零，即将卡号域置空，属无效的情况。其实，我们已经在非数字的划分中提到过这种情况，不过这没什么。我们设计出重复的输入值总比完全漏掉的好。在

实际测试的时候没必要真的去测试两次。

2. 卡号太短，属无效情况。对于大多数信用卡我们会测试 1 位卡号和 15 位卡号的情况，而对于 American Express 我们需要测试 14 位卡号，因为对于这个类型的信用卡的卡号有效位数是 15 位。

3. 卡号的长度适当，属有效情况（至少从输入域的角度来讲）。我们在稍后会讲到如何验证卡号和其他域是否真正有效。

4. 卡号太长，属无效情况。对于大部分信用卡我们将测试 17 位卡号的情况，而对于美国运通信用卡，我们会测试 16 位的卡号，因为对这种信用卡，15 位卡号是合法的。同时我们还应测试一些最大输入的情况。

您可能会觉得我们为了一个信用卡输入域做这么多分析好像有点可笑或者有点做过头了。然而，从安全的角度来讲，您应该对这个输入界面进行大量测试，不是吗？风险分析告诉我们确实应该这么做。

由于篇幅有限，本书中我没有对信用卡安全码、有效期截止的月份和年份，或持卡人姓名进行同样的（等价类和边界值）分析。对信用卡安全码进行等价类划分与对卡号的划分非常类似，包括美国运通信用卡的安全码的位数与其他信用卡也是不同的。

确定信用卡有效期截止的月份和年份也是比较简单的，因为 RBCS 网站的设计者故意将这两个域限制成只能使用下拉菜单。这提高了可测试性，因为人们无法输入月份 Februly 或年份 20ZB。

对于持卡人姓名这个域就有点复杂了，因为我们必须知道域的最大长度和最小长度，以及什么样的字符规则是有效的？若有相应的需求规格说明，那么会对我们（进行相关测试用例设计）有很大帮助。

当然，我们上面做的这些工作只是对单个域进行的验证。一旦这些单个验证完成以后，系统会将这些支付方式相关的信息发送到信用卡所在公司进行验证。我们可以对这些信息的整体进行等价类划分，不过实际上这个过程取决于一系列的条件：

- 持卡人是否是该信用卡的真正持有者，其他相关信息正确吗？
- 这个信用卡是有效的还是被注销了？
- 正进行的交易是否在持卡人的限额以内还是超过了限额？
- 是从一个普通的站点还是从一个可疑的站点提交了这个交易？

在后续小节中，当我们讲解决策表的时候，会提到如何对组合的条件进行测试。

4.2.8 有多少边界值

接下来我们讲解边界值分析的最后一个问题。若您不曾阅读一些测试书籍的话，您可能没有想到过会存在这个问题：在边界上有多少边界值？

到目前为止，我说过每个边界有两个边界值。边界位于一个等价类的最大值和上一级等价类的最小值之间。也就是说，边界本身并不属于任何等价类。

但是，有些作者说有 3 个边界值。伯利兹-贝哲可能是其中最有名的一个。那么，为什么会有 3 个边界值呢？

pin51.com 最新最全软件资源分享 拼吾爱

主要的区别在于这里要使用数学分析而不是我先前用的图形分析。在图 4-20 中您可以看到两个数学不等式及其图形模型。根据贝哲规则，您可以选择边界值自身作为中间边界值，然后在这个基础上增加或减去最小的可行数目，就生成了另外两个边界值。在这个例子中，最小的可行数目是一，但对于浮点数来说就很麻烦了，我们必须考虑精度问题。

正如您在图 4-20 中所看到的那样，数学分析产生的边界总是包含了图形分析产生的边界。在我看来花这么多时间去设计这 3 个边界值就是浪费时间，还不如设计更多的测试用例去覆盖其他部分，尤其是那些未覆盖的风险。

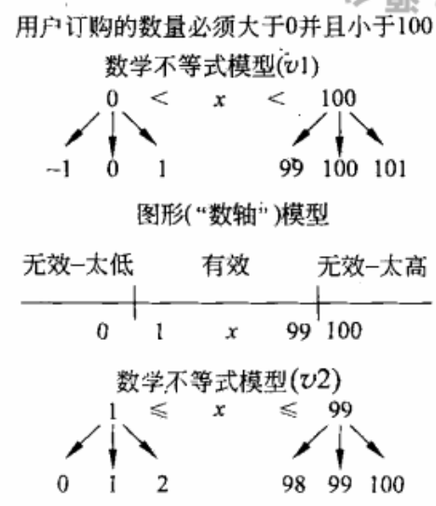


图 4-20 两个还是 3 个边界值？

4.2.9 边界值练习

图 4-21 是 HELLOCARMS 系统的某个界面的原型。在这个界面中，系统要求用户输入两个部分的信息：

- 贷款额度。
- 房产价值。

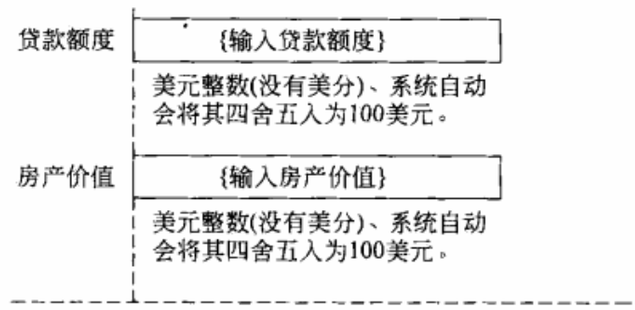


图 4-21 HELLOCARMS 系统中某个账户的界面原型

对于上述两个域，系统只接受美元整数输入（不接受美分），然后系统会将这个输入值取整为最接近的 100 美元。

假设对于贷款有如下规则：

- 最小贷款额度为 5 000 美元。
- 最大贷款额度为 1 000 000 美元。
- 最小房产价值为 25 000 美元。
- 最大房产价值为 5 000 000 美元。

请参考附录 B HELLOCARMS 系统需求文档中“系统功能需求”部分的 010-010-130 项以及 010-010-140 项。

若这两个域有效，那么系统就会接受输入。而后电话银行经理将会处理用户申请或者这份申请会被移交给高级电话银行经理进行处理。若这两个域有一个无效或都无效，那么系统会返回一条错误消息。

练习包括两个部分：

1. 对这两个域进行等价类划分和边界值分析，并指出其中的有效等价类和无效等价类，以及这些等价类的边界值。

2. 设计测试用例以覆盖这些划分和边界值，记住要将无效等价类和有效等价类组合在一起测试。

这两份练习的答案在下面，在和自己的答案进行对照的时候，建议您在看第二部分的答案之前，应该先看看第一部分的答案，以和自己的答案对照（如果有必要，更改第二部分的答案）。

4.2.10 边界值练习解答

首先，在图 4-22 中是等价类划分和边界值分析的结果。

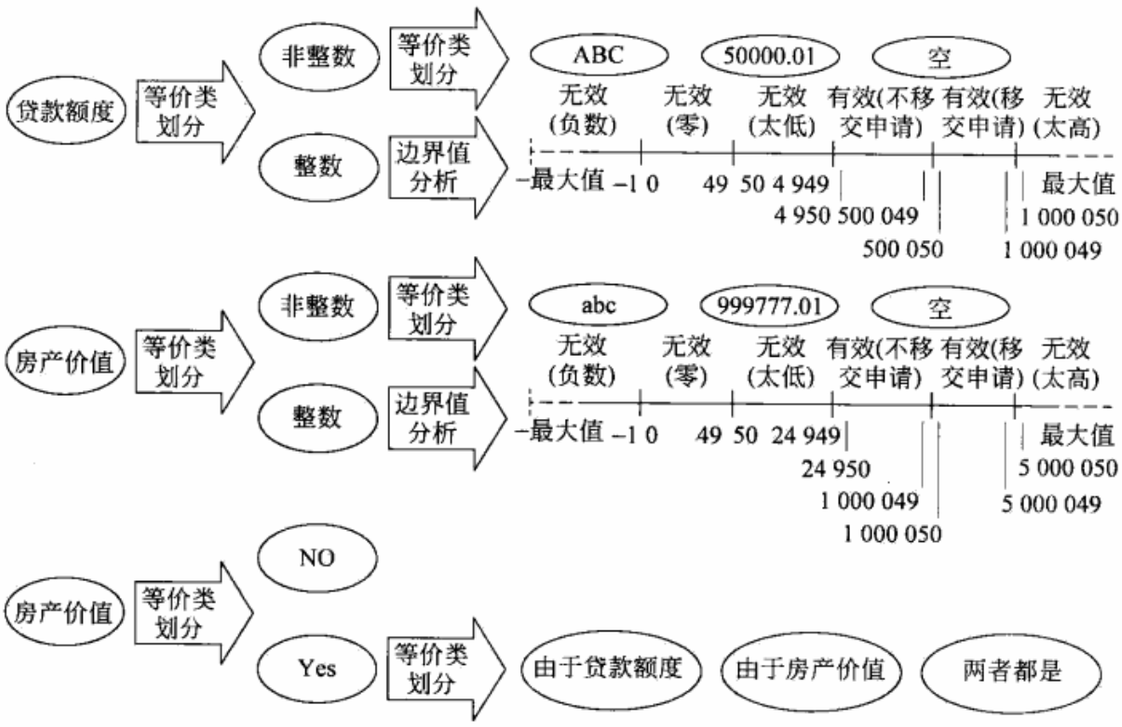


图 4-22 对于贷款额度、房产价值和移交申请进行等价类划分和边界值分析

因此，对贷款额度进行边界值分析和等价类划分的结果如下：

#	划　分	边 界 值
1	字符：ABC	—
2	小数：50 001.01	—
3	空	—
4	无效（负数）	负最大值
5	无效（负数）	-1
6	无效（零）	0
7	无效（零）	49

pin51.com 拼吾爱 最新最全资源分享下载

续表

#	划　分	边 界 值
8	无效（太小）	50
9	无效（太小）	4 949
10	有效（不用移交申请）	4 950
11	有效（不用移交申请）	500 049
12	有效（移交申请）	500 050
13	有效（移交申请）	1 000 049
14	无效（太大）	1 000 050
15	无效（太大）	最大值

对房产价值进行等价类划分和边界值分析的结果如下：

#	划　分	边 界 值
1	字符：abc	—
2	小数：999 777.01	—
3	空	—
4	无效（负数）	负最大值
5	无效（负数）	–1
6	无效（零）	0
7	无效（零）	49
8	无效（太小）	50
9	无效（太小）	24 949
10	有效（不用移交申请）	24 950
11	有效（不用移交申请）	1 000 049
12	有效（移交申请）	1 000 050
13	有效（移交申请）	5 000 049
14	无效（太大）	5 000 050
15	无效（太大）	最大值

对于移交申请的决策，我们可以根据贷款额度和房产价值进行如下划分：

#	划　分
1	否
2	是，由于贷款额度
3	是，由于房产价值
4	是，两个都是

现在，让我们设计测试用例，覆盖这些等价类划分和边界值。在这个过程中，我将

提供相关信息以保证测试用例号与等价类划分之间的可追溯性。就像以前的例子中提到过的，一旦每一个划分都有一个对应的测试用例，就表明我已经完成了工作——只要我没去将无效划分组合在一起就行了！

输　入	1	2	3	4	5	6
贷款额度	4 950	500 050	500 049	1 000 049	ABC	50 000.01
房产价值	24 950	1 000 049	1 000 050	5 000 049	100 000	200 000
输　出						
接受？	是	是	是	是	否	否
移交申请？	否	是（贷款）	是（房产）	是（都是）	—	—

输　入	7	8	9	10	11	12
贷款额度	空	100 000	200 000	300 000	负最大值	–1
房产价值	300 000	abc	999 777.01	空	400 000	500 000
输　出						
接受？	否	否	否	否	否	否
移交申请？	—	—	—	—	—	—

输　入	13	14	15	16	17	18
贷款额度	0	49	50	4949	1 000 050	最大值
房产价值	600 000	700 000	800 000	900 000	1 000 000	1 100 000
输　出						
接受？	否	否	否	否	否	否
移交申请？	—	—	—	—	—	—

输　入	19	20	21	22	23	24
贷款额度	400 000	500 000	600 000	700 000	800 000	900 000
房产价值	负最大值	–1	0	49	50	24949
输　出						
接受？	否	否	否	否	否	否
移交申请？	—	—	—	—	—	—

输　入	25	26	27	28	29	30
贷款额度	1 000 000	555 555				
房产价值	5 000 050	最大值				
输　出						
接受？	否	否				
移交申请？	—	—				

pin51.com 拼吾爱 最新最全源分享下载站

贷款额度：

#	划　分	边 界 值	测试用例
1	字符：ABC	—	5
2	小数：50 001.01	—	6
3	空	—	7
4	无效（负数）	负最大值	11
5	无效（负数）	–1	12
6	无效（零）	0	13
7	无效（零）	49	14
8	无效（太小）	50	15
9	无效（太小）	4 949	16
10	有效（不用移交申请）	4 950	1
11	有效（不用移交申请）	500 049	3
12	有效（移交申请）	500 050	2
13	有效（移交申请）	1 000 049	4
14	无效（太大）	1 000 050	17
15	无效（太大）	最大值	18

房产价值：

#	划　分	边 界 值	测试用例
1	字符：abc	—	8
2	小数：999 777.01	—	9
3	空	—	10
4	无效（负数）	负最大值	19
5	无效（负数）	–1	20
6	无效（零）	0	21
7	无效（零）	49	22
8	无效（太小）	50	23
9	无效（太小）	24 949	24
10	有效（不用移交申请）	24 950	1
11	有效（不用移交申请）	1 000 049	2
12	有效（移交申请）	1 000 050	3
13	有效（移交申请）	5 000 049	4
14	无效（太大）	5 000 050	25
15	无效（太大）	最大值	26

ptpress.com.cn 异步图书 提供下载

移交申请决策：

#	划　分	测 试 用 例
1	否	1
2	是，由于贷款额度	2
3	是，由于房产价值	3
4	是，两个都是	4

注意，在我们设计测试用例的时候覆盖到了另一种值得关注的关于移交申请决策的组合，即当输入值被系统拒绝的时候，我们应该无法进入下一个界面，更不可能发生移交申请的情况了。除此之外，我们也设计了这样的测试：贷款额度或房产价值已经满足了触发移交申请的规则要求，但由于另一个输入值无效，系统拒绝接受输入，也不会移交申请。其实我们也可以将这种情况作为移交申请决策期望值为“有效接受”、“无效不接受”之外的第三组等价类。即，一个输入有效值触发了移交申请，但另一个输入值无效。它的两个子等价类为“贷款额度触发移交申请，但房产值无效”，和“房产值触发移交申请，但贷款额度无效”。

4.2.11 决策表

等价类划分技术和边界值分析技术都是非常有用的技术。正如我们在本节前面一些部分看到的那样，这两种技术对于测试用户界面的输入域验证特别有用。然而，作为测试分析师而言，我们做得很多测试针对的是用户界面这一层面之下的业务逻辑。虽然我们也可以使用等价类划分技术和边界值分析技术来测试业务逻辑，但是另外 3 种技术，决策表、基于用例和基于状态的测试，在测试业务逻辑的时候通常显得更容易也更高效。让我们先来看看决策表。概念上来讲，决策表表示的是系统处理交易情况（transactional situation）时所用的规则。由于决策表简单、明了，因此特别适用于测试那些规则，一般我们会为每条规则设计至少一个测试用例。

ISTQB 术语

因果图：用来表示输入（原因）与结果之间关系的图表，因果图可以用来设计测试用例。

因果图技术：通过因果图（case-effect graph）设计测试用例的一种黑盒测试设计技术。

上面我所说的“交易情况”是指，在某一指定时刻的单笔交易的一些条件，例如输入、前置条件等，足够触发系统做出相应的响应的情况。如果当前条件不够，但我们还有必要参考曾经有过什么条件被满足而触发系统响应，那么这种情况下我们应该用基于状态的测试技术，我们将在其后章节讨论这种技术。

决策表所使用的模型通常是一个表，不过有时候也会用布尔图形。无论哪种模型，都包括了两部分：各种条件的组合以及在满足条件组合的情况下系统的响应或操作。

pin51.com 拼吾爱 最新最全

布尔图的正式名称应该叫做因果图。不过，这不重要，重要的是我们要记住因果图和决策表之间是可以互相转换的。因此这两个模型随便您使用哪个都可以。我倾向于决策表，它们更常用，因此我在这里主要讲决策表。不过稍后我会教您怎么进行决策表和因果图之间的转换。

若要根据决策表或因果图设计测试用例，我们应该设计测试输入以满足给定的条件。测试输出则应与条件组合对应的操作相一致。在测试执行过程中，我们检查系统实际执行的操作或输出与期望的操作或输出是否一致。

我们应该创建足够多的测试用例，确保每个条件组合至少被覆盖到了一次。不过我们经常会放宽这个覆盖率准则，比如改成我们必须覆盖那些能决定操作的条件组合。若您没有很好地为基础级认证考试做准备，这里您可能觉得有点糊涂，的确是这样的，因为在大多数书本或课程都没有充分讲解这两者的区别。当讲到收缩决策表（collapsed decision table）的时候您就会明白这两种覆盖率之间的区别了。

对于决策表，我们将覆盖率准则归结为一个简单的规则，即表中的每一列至少要有一个测试用例。而对于因果图，您必须先要根据图形生成一个所谓的“真值表”，在这个表中包含了所有可能的条件组合，然后在设计测试用例时确保真值表中的每行都有一个对应的测试用例。

那么，对于决策表我们做了什么样的缺陷假设呢？我们在找什么类型的缺陷？有两种。第一种，在某些条件组合的情况下，系统会采取错误操作。也就是说，在这种条件组合情况下，系统执行了不应该执行的操作。第二种，在某些条件组合情况下，系统没有执行正确的操作。也就是说，在这种条件组合情况下，系统没有执行它应该执行的操作。

我们曾提到过 RBCS 网站（www.rbcs-us.com）的电子商务模块，并学习了如何去验证支付信息，即信用卡类型、卡号、信用卡安全码、过期月份、过期年份以及持卡人姓名。当时的目标是使用等价类划分技术和边界值分析，测试系统是否能在将信息提交给服务器之前正确地验证各种支付信息。

那么，我们应该如何测试信用卡处理公司是否能正确验证这些信息？当然，我们还可以使用等价类划分技术，不过实际上这个过程取决于一组条件：

- 持卡人是否是该信用卡的真正持有者，其他的信息正确吗？
- 这个信用卡是有效的还是被注销了？
- 正进行的交易是否在持卡人的限额以内还是超过了限额？
- 是从一个普通的站点还是从一个可疑的站点提交了这个交易？

根据图 4-23 中的决策表可以知道，这 4 个条件决定了系统会执行 3 个操作中的哪个操作：

- 我们是否应该批准这个交易？
- 我们是否应该联系持卡人（例如，提示他们该信用卡在陌生的地方进行了消费）？
- 我们是否应该联系开户银行（例如，要求他们没收这张已注销的信用卡）？

让我们仔细瞧瞧这个决策表有些什么。在表的上方列出了条件，而下方是相应的操作。除了最左边的一列外，其他每列都代表了一条业务规则。大致上，每条规则的意思

条件	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
真实的账户	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
有效的账户	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
在限额之内	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
交易站点是否正常	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
操作										?						
批准交易	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
通知持卡人	N	Y	Y	Y	N	Y	Y	Y	N	N	N	N	N	N	N	N
通知开户银行	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

图 4-23 决策表示例（完整）

是“在这种条件组合的情况下（如每列的上方所示），执行这些操作（如每列的下方所示）。”

注意，表中的列数，或者说业务规则的数目，应该等于 2 的 n 次方，n 等于条件的数目。也就是说，2 乘以 2 乘以 2 乘以 2 等于 16。当这些条件是严格的布尔类型，即其值只能是“是”或“否”，并且这个决策表就是一个完整的决策表（不是一个收缩的决策表）时，表中的列数会满足这种规则。

您是否注意到我是如何在表中填入条件的？最上方的条件变化得最慢。有一半的列填“是”，另一半的列填“否”。第二个条件比最上方的条件变化得快一点，不过比其他的条件慢。其中 1/4 填“是”，1/4 填“否”，然后再是 1/4 填“是”，1/4 填“否”。如此类推，对于最下方的那个条件，应该填“是”，“否”，“是”，“否”，“是”，等等。这种模式能确保您不会有遗漏。若您从最上方的条件开始填，在左半部分的列中填“是”，右半部份的列中填“否”，然后参照我所说的模式进行填写，最后在最下方的条件中填“是”，“否”，“是”，“否”，“是”，等等，如果其中哪个条件没有按照这个模式做，那么就存在错误。

根据这个决策表设计测试用例很方便：每一列对应一个测试用例。当实际运行测试的时候，我们根据条件设计测试输入。我们将值为“是/否”的条件替换成信用卡号、信用卡安全码、过期日期和持卡人姓名域的实际输入。然后我们在执行测试的时候检查系统是否执行了预期的操作。

在某些情况下，对于决策表中的每一列，我们需要设计不止一个测试用例。稍后会讲到这点，我们将使用先前学的测试技术，等价类划分技术和边界值分析技术，拓展决策表测试。

4.2.12 在表格中收缩列项

您也许会注意到，有些测试用例并没有多少意义。例如，一个账户怎么可能既是不存在的又是有效的？或既是不存在的又是在限额内的？这通常意味着在决策表内并不是每一列都是有意义的。

有时候，我们通过合并列项收缩决策表，使得决策表能更简洁，更有意义。当决策表内的某项或多项条件的值的变化并不会影响相关条件组合所对应的操作时，我们可以收缩这个决策表。

pin51.com 最新最全软件测试资料 拼吾爱

ISTQB 术语

决策表测试：一种黑盒测试设计技术，设计的测试用例用来测试决策表中各种输入条件和/或触发的原因的组合。

正如我所说的那样，若一些条件并不会影响操作，那么我们可以将两列或多列合并在一起。通常来说，可合并的列都是相邻的。虽然这并不是绝对的，不过至少您在合并的时候可以先看看相邻的列是否能合并。

在合并列的时候，应该找那些操作完全相同的列。注意，此处所指的操作相同是针对决策表中所有的操作而言的，而不是其中的一部分。找到这样的列之后，您会发现这些列的某些条件是相同的，某些条件是不同的。显然，那些不相同的条件并不会影响输出。接下来，我们将这些列中的不同条件用破折号代替。破折号通常意味着我不在乎、它不会影响结果或者在给定条件下这件事不可能发生。

不断重复上述过程，直到您已经将所有要合并的列找出，并将这些列中无关紧要的条件的值设置为“-”。如果您还是不太明白，没关系，在接下来的示例中我会进一步讲解的。

有一点需要注意的是，当您遇到一个决策表，在表中有不止一条规则适用时要小心处理。这些表中的规则不是排他性的。稍后我们在本节中还会提到这点。

图 4-24 与图 4-23 是同一个决策表，但经过了收缩，消除了多余的列。最明显的地方是原先的第 9 到第 16 列合并成了一列。

条件	1	2	3	5	6	7	8
真实的账户	Y	Y	Y	Y	Y	Y	N
有效的账户	Y	Y	Y	N	N	N	–
在限额之内	Y	Y	N	Y	Y	N	–
交易站点是否正常	Y	N	–	Y	N	–	–
操作							
批准交易	Y	N	N	N	N	N	N
通知持卡人	N	Y	Y	N	Y	Y	N
通知开户银行	N	N	N	Y	Y	Y	Y

图 4-24 决策表示例（收缩的）

为了方便大家对照着阅读，我并未改动列号。现在来看看我是怎么进行合并的。仔细观察 1、2、3 列。注意，我们不能将第 2 和第 3 列合并，因为这么做会使得“限制之内”和“地点确认”这两个条件都变成“—”，而如果您仔细研究过图 4-24 或图 4-23 之后，您会发现若要触发“联系持卡人”这个操作，则这两个条件中有一个必须是“否”。第 3 列与第 4 列合并意味着，如果交易超过了信用卡的限额，那么系统不再确认地点，而是直接通知持卡人。第 7 列，第 8 列的合并也是同样的道理。

注意图 4-24 中决策表的格式并没有变。条件在表的左上角，操作在表的左下角。除了最左边的那一列，其他每一列都代表一条业务规则。每条规则的意思是“在这种条件组合的情况下（如每列的上方所示，某些条件可能无关紧要），执行这些操作（如每列的

下方所示，所有操作都是明确的）。

经过收缩的决策表不再符合“列数等于 2 的 n 次方，n 为条件数”这一规律。因为若要遵守这条规律就没法对决策表进行收缩了。如果您怕漏掉一些重要的东西，那么每次设计决策表的时候可以先从完整的决策表开始，因为这样就能保证包含了所有的条件组合。您可以用数学方法检查决策表是否完整。然后，再仔细地收缩决策表，减少要设计的测试用例的数目。

同样需要注意的是，当您收缩了决策表之后，原先完整表中那种好用的“是、否、是、否”的排列模式也没有了。这也是为什么您要小心翼翼地收缩决策表的原因之一，因为您不能再通过这种模式或数学方法去检查是否有所遗漏。[3]

4.2.13 因果图

当您收缩决策表的时候，因果图能确保您不会无意间合并了不恰当的列。有些人喜欢直接使用因果图进行测试设计，不过，我想我已经提到过，我不赞成这样做。

图 4-25 列出了决策表和因果图之间互相转换的主要过程。我们先看左半部分。

决策表转换成因果图	因果图转换成决策表
— 在空白页的左侧列出所有条件	— 在一张空白决策表的左上方列出所有的条件
— 在空白页的右侧列出所有操作	— 在条件的下方，即表的左下方列出所有的操作
— 观察决策表中的条件组合是如何触发每个操作的	— 列出所有可能的条件组合
— 用布尔操作符将一个或多个条件与每个操作连接在一起	— 对照因果图，确定对于每一种条件组合操作是否会发生
— 对决策表中的每一个操作重复这个操作	— 一旦您确定了操作部分所有的值，您就可以开始收缩决策表

图 4-25 转换成/自因果图

根据决策表创建因果图的方法是，首先在一张白纸的左边列出所有的条件。接下来，在纸张的右半部分列出所有的操作。如果有很多条件和操作，那么显然这里您需要一张大纸来画因果图，不过通常这种情况表明您的决策表本身也很大。

接下来，仔细观察决策表中的条件组合是如何触发每个操作的。如图 4-26 所示，用布尔操作符将一个或多个条件与每个操作连接在一起。对决策表中的每一个操作重复这个过程。

如果您手上正好有一个因果图，然后想要创建一个决策表，那么首先您应该在一张空白决策表的左上方列出所有的条件。然后，在条件的下方，即表的左下方列出所有的操作。使用我们先前提到的模式，列出所有可能的条件组合。再对照因果图，确定对于每一种条件组合操作是否会发生。一旦确定了操作部分所有的值，就可以开始收缩决策表。

在图 4-26 中可以看到与我们刚刚使用的决策表相对应的因果图。您可能会问，“对应于哪一个决策表，完整的那个还是收缩的那个？”都是。除非收缩过程中出了问题，

3 如果您希望查阅与决策表相关的主题，包括完整的和收缩的决策表，可以参考我的书 Pragmatic Software Testing 以及李·科普兰（Lee Copeland）的书 The Practitioner's Guide to Software Test Design。

pin51.com 最新最全源码分享下载站 拼吾爱

否则这两个决策表应该是逻辑对等的。

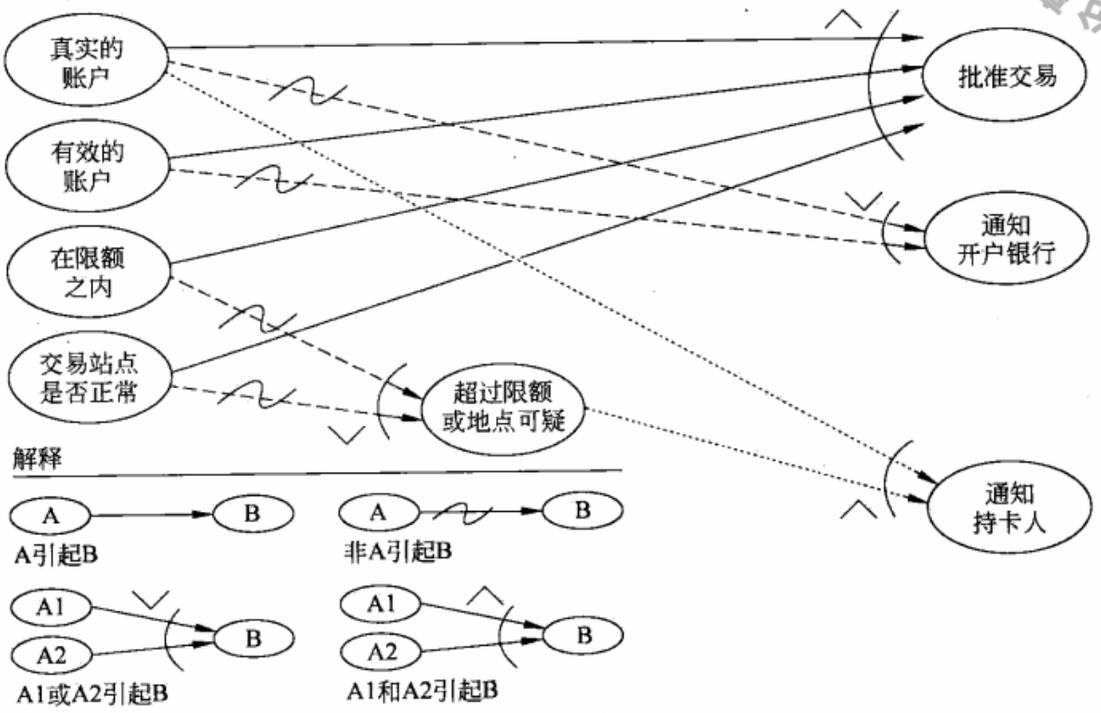


图 4-26 因果图示例

可以看到在图的左下方有一份说明，告诉您如何去理解这些运算。让我们从这份说明的左上方开始，按顺时针方向，逐个介绍这些运算。

简单的因果关系：如果 A 为真，那么 B 为真，或者说，A 触发 B。

否定关系：如果 A 为假，那么 B 为真，或者说，假的 A 触发 B。

逻辑与运算：如果 A1 和 A2 都为真，那么 B 为真，或者说，A1 和 A2 触发 B。

逻辑或运算：如果 A1 或 A2 为真，那么 B 为真，或者说，A1 或 A2 触发 B。

现在让我们来看看条件和操作怎么连接在一起。表示因果关系的实线和逻辑与运算符组合在一起，表示必须满足所有 4 个条件系统才会批准这项交易。

带有否定操作符的表示因果关系的虚线以及一个逻辑或操作符，表示如果账户不存在或账户不是有效的，那么系统会联系开户银行。

表示因果关系的点划线稍许复杂一点。首先，我们用否定操作符和逻辑或操作符将“限额之内”和“地点确认”这两个条件组合在一起，生成一个中间条件“超过限额或地点可疑”。然后我们将这个中间条件和条件“真实的信用卡”组合在一起，表示如果信用卡账户是真实的，但交易超过了限额或交易地点比较可疑，那么系统会联系持卡人。

4.2.14 将决策表测试与其他技术结合

我曾经提到过，若将等价类划分技术和决策表技术组合在一起使用，那么表中的每一列就可能会有多个测试用例。现在让我们回到图 4-23 中提到的决策表示例的第 9 列。

我们可以对这个问题进行等价类划分：“从测试的角度来看，有多少种方法可能使得一个账户是不真实的？”正如图 4-27 所示，有 5 种潜在的方法：

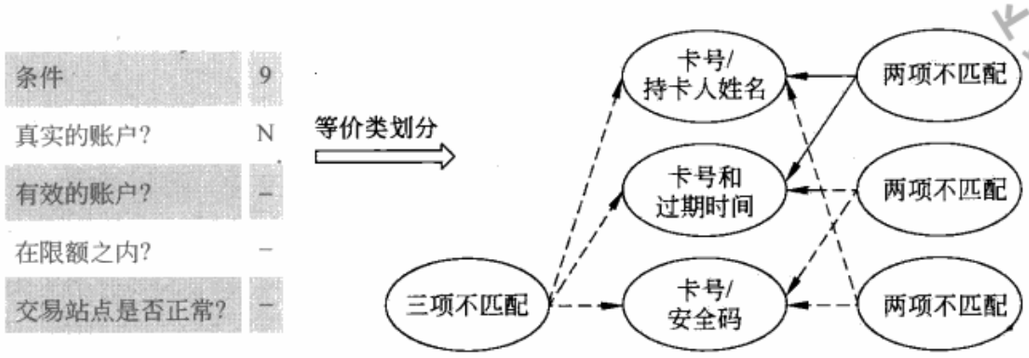


图 4-27 等价类划分和决策表

- 卡号与持卡人不匹配。
- 卡号与过期时间不匹配。
- 卡号与安全码不匹配。
- 以上 3 项中有两项同时不匹配（3 种可能）。
- 3 种都不匹配。

因此针对这一列应该有 7 个测试用例。

那么边界值分析呢？的确，我们也可以在决策表测试中应用边界值分析以设计新的有价值的测试用例。例如，有多少与信用卡额度相关的有价值的测试值？

如图 4-28 所示，通过使用等价类划分和边界值分析技术，我们得到了 6 种可能性：

- 账户内初始余额为零。
- 交易结束后账户内余额处于正常状态。
- 交易结束后该账户恰好用完所有信用额度。
- 交易结束后账户会恰好超出信用额度。
- 交易之前账户处于恰好用完信用额度状态（这可能会使该账户在交易结束后受到全面的审查）。
- 交易结束后该账户将处于最大透支额状态（这种情况几乎不可能出现）。

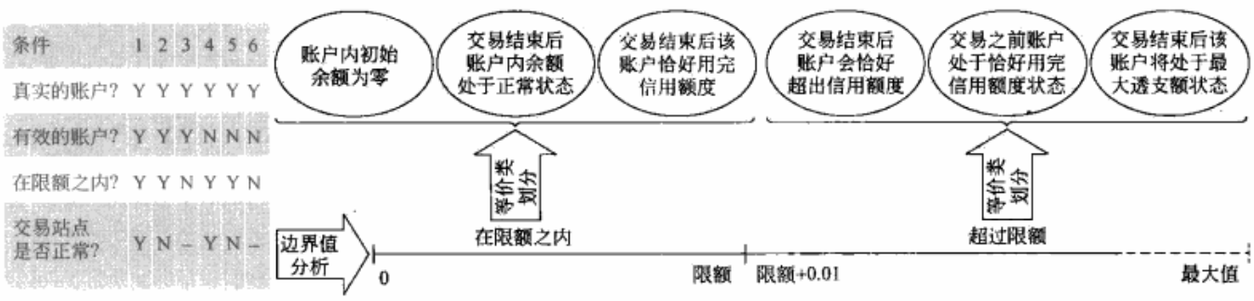


图 4-28 边界值和决策表

将这 6 种可能性与原先的决策表相结合之后，我们可预见到需要设计比原计划的“一列一个测试用例”更多的“超过限额”的测试用例，因此总体来说测试用例只是略有增加。换句话说，（针对图 4-24）我们将设计 4 个限额之内的测试用例和 3 个超过限额的测试。不过如果您希望在被批准的交易中测试每一个限额之内的等价类，那么情况就不一样了，在这种情况下，第 1 列将从 1 个测试用例变成 3 个测试用例。

pin51.com 最新最全源分享下载 拼吾爱

4.2.15 决策表中的非排他性规则

让我们来讨论一下早些时候我曾经提到过的决策表中的非排他性规则。

有时候对于一个交易可能有不止一条规则适用。在图 4-29 中，可以看到一个关于信用卡费用计算的决策表。表中共有 3 个条件，注意，在某一个月份中，有可能零个、一个、两个或者所有这 3 个条件都满足。这种情况对我们的测试有什么影响呢？这会将测试变得稍微复杂，不过我们可以使用数学方法和基于风险的测试来避免遗漏一些主要的隐患。

条件	1	2	3
外汇?	Y	–	–
余额移后?	–	Y	–
延迟付款?	–	–	Y
操作			
交易费用?	Y	–	–
收取利息?	–	Y	–
收取滞纳金?	–	–	Y

图 4-29 非排他性规则示例

在一开始时，将这个决策表当成普通的决策表进行测试，每次只使用一条规则，确保您没有满足与该规则不相关的条件。这种方法能让您在封闭的环境中测试各个规则，就好像在测试排他性规则一样。

接下来，考虑测试规则的组合。注意，我说的“考虑”并不是要您“测试所有可能的规则组合”。当测试人员不考虑测试用例的价值，想要测试所有因素的组合时，测试用例的数目会爆炸式地激增，应该要避免这种情况。在本示例中，一共只有 8 种可能的组合——3 个因素，每个因素有 2 个可选项，2 乘以 2 乘以 2 等于 8。然而，如果您有 6 个因素，每个因素有 5 个选项，您会有 15 625 个组合。

有一种避免组合爆炸的方法是识别可能的组合，并根据风险将其排序。只测试其中重要的组合，而忽略其余的部分。

另一种避免组合爆炸的方法是使用分类树和结对测试技术，我们稍后会讲到这两种技术。[4]

4.2.16 决策表练习

在 HELLOCARMS 项目的开发过程中，项目团队为 HELLOCARMS 系统添加了一项新的功能。这项功能允许系统向用户销售一份人寿保险合同，这份保险的赔偿金足以偿还房屋净值贷款，因此如果借款人身亡，保险公司会为其偿还贷款。在保险期间的每一年年初，保户都要根据自己当前的贷款余额计算该年度的保费。基本保费的计算方法是

4· 我在 Pragmatic Software Testing 一书中详细讲解了有非排他性规则的决策表这一主题。

贷款余额中每 10 000 美元支付 1 美元保费。该保险政策不适用于已申请到房屋净值信贷额度和反向住房按揭贷款的客户。

除了基本保费之外，电话银行经理会在面试时询问用户一些基本的身体和健康方面的问题，然后系统再根据面试的结果在基本保费的基础上增加一定百分比的费用。

若用户对下列问题中任一问题的回答是“是”，那么他的保费将在基本保费的基础上增加 50%。

1. 您在过去的 12 个月里抽过烟吗？
2. 您是否曾经被诊断出有癌症、糖尿病、胆固醇过高、高血压、心律失常或中风？
3. 除了生小孩或骨折之外，在最近 5 年内您是否曾住院超过 72 小时？
4. 在最近的 5 年内，您是否曾因为受伤或某种疾病而无法工作超过一周？

电话银行经理也会询问年龄、体重和身高。（申请人年龄不能低于 18 岁）然后将体重和身高组合在一起计算体质指数（BMI）。电话银行经理会根据体质指数以及表 4-1 来确定保费增加的百分比或甚至会拒绝有问题的用户参保，其中表 4-1 代表此人在未来患体重相关疾病的可能性。

表 4-1 体质指数/年龄 保费增长百分比

	体质指数（BMI）			
年龄	<17	34～36	37～39	>39
18～39	拒绝	75%	100%	拒绝
40～59	拒绝	50%	75%	拒绝
>59	拒绝	25%	50%	拒绝

保费的增长是累积性计算的。例如，一个正常体重的人，抽烟并有高血压，那么他每年的保险费用由每 10 000 美元交 1 美元增加到每 10 000 美元交 2.25（1*1.5*1.5）美元。如果是一位 45 岁的男性糖尿病患者，且他的体质指数为 39，那么他每年的保险费用由每 10 000 美元交 1 美元增加到每 10 000 美元交 2.625（1*1.5*1.75）美元。

练习包括 4 个步骤：

1. 根据 4 个健康问题、体质指数以及它们各自的效果设计一个决策表。
2. 将决策表转换成因果图。
3. 列出体质指数和年龄的边界值。
4. 设计测试用例覆盖决策表和边界值，记得使用测试非排他性规则的方法。

这 4 部分的答案在下面。在您查看下一部分的答案前，应该先查看前一部分的答案，以和您自己的答案对照（并且，有必要的话，修改您自己下一部分的答案）。

注意：取决于您自己的偏好，您可以交换第 1 步和第 2 步的顺序。

4.2.17 决策表练习解答

首先，我根据 4 个健康问题和体质指数/年龄表格设计了决策表，如表 4-2 所示。注意其中保费的增长是以百分比计算的。

pin51.com 最新最全软件测试教程 拼吾爱

表 4-2 人寿保险费用增长率决策表

条 件	1	2	3	4	5	6	7	8	9	10	11	12
吸烟?	是	—	—	—	—	—	—	—	—	—	—	—
诊断?	—	是	—	—	—	—	—	—	—	—	—	—
住院?	—	—	是	—	—	—	—	—	—	—	—	—
伤残?	—	—	—	是	—	—	—	—	—	—	—	—
体质指数	—	—	—	—	34～36	34～36	34～36	37～39	37～39	37～39	<17	>39
年龄	—	—	—	—	18～39	40～59	>59	18～39	40～59	>59	—	—
操作												
费用增长	50	50	50	50	75	50	25	100	75	50	—	—
拒绝	—	—	—	—	—	—	—	—	—	—	是	是

很重要的一点是规则 1 到规则 4 是非排他性的，而规则 5 到规则 12 是排他性的。

另外，有一个隐含的规则，即申请人必须大于 17 岁，不然他不只无法参保，连贷款本身也是不行的。我可以将这一条放在决策表中，不过由于我主要是在进行业务功能的测试，而不是测试输入验证，所以并没这么做。我会用边界值来覆盖这些测试。

接下来，我画出了对应的因果图，如图 4-30 所示。

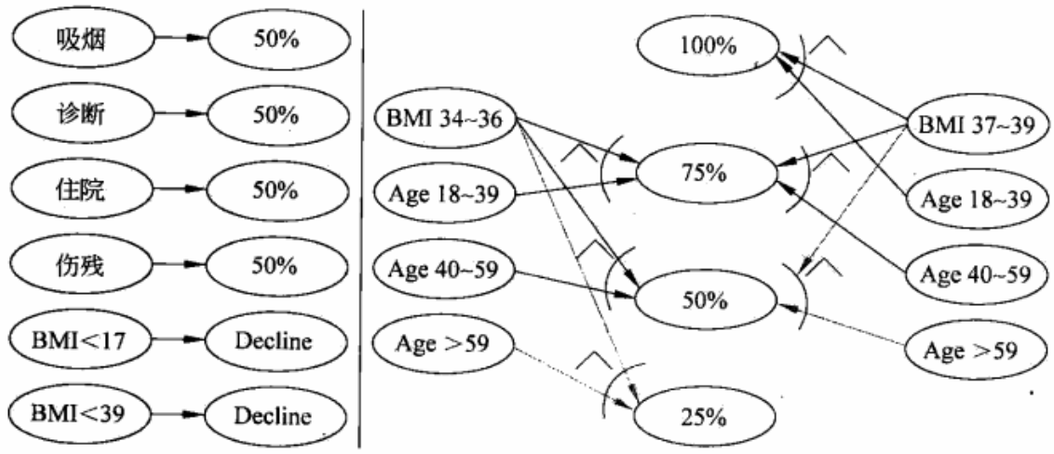


图 4-30 人寿保险因果图

注意图的左侧，各条非排他性规则是单独列出的，而不是将它们用逻辑或运算符组合起来从而触发 50%这个操作。为什么？因为如果我这么做，那么保费的增长就不再是累积性的了。我在图中也列出了两种会导致拒绝操作的体质指数条件。

现在，让我们来看看体质指数和年龄的边界值，如图 4-31 所示。

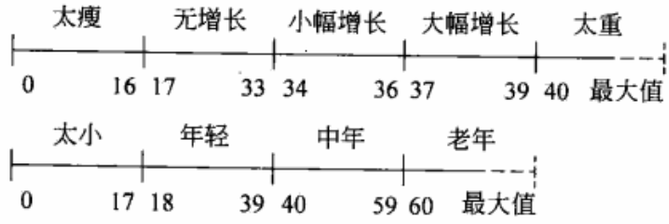


图 4-31 体质指数和年龄的边界值

有 3 个关于体质指数的测试要点。第一点，体质指数不是直接输入的，而是根据输入的身高和体重计算出来的。根据这两个域的范围和精度，对于一个给定的体质指数，我们可以有很多种输入组合。第二点，通过输入最小身高以及最大体重，我们可以得到体质指数的最大值。第三点，我们要独立地分析这两个域的边界值并确保对这些边界值进行适当地测试。

关于年龄的一个测试要点是：您可以看到我忽略了对无效年龄的划分，比如负年龄和在年龄域中输入非整数。再一次说明，我的意思是我们要将此处的测试与测试输入域的验证过程分开。在这里，我们的重点是测试业务逻辑。

最后一点，对于体质指数域和年龄域，我都未指出它们的最大值。要么在测试设计过程中，有人会给我一份测试规格说明，在上面会指出这两个最大值，或者在测试执行过程中我会根据以往的经验确定这两个最大值。

对于体质指数，其边界值和等价类如下所示：

#	划　分	边 界 值
1	太瘦	0
2	太瘦	16
3	无增长	17
4	无增长	33
5	小幅增长	34
6	小幅增长	36
7	大幅增长	37
8	大幅增长	39
9	太重	40
10	太重	最大值

对于年龄，其边界值和等价类如下所示：

#	划　分	边 界 值
1	太小	0
2	太小	17
3	年轻	18
4	年轻	39
5	中年	40
6	中年	59
7	老年	60
8	老年	最大值

最后，给出我设计的测试用例。这些表格和决策表很像，不过请注意，这里我直接给出最后的费用（贷款余额中每 10 000 美元需要支付的保险费用），而不是费用增长的百分比。

pin51.com 最新最全软件资料 拼吾爱

	测试用例											
条件	1	2	3	4	5	6	7	8	9	10	11	12
吸烟?	是	否	否	否	否	否	否	否	否	否	否	否
诊断?	否	是	否	否	否	否	否	否	否	否	否	否
住院?	否	否	是	否	否	否	否	否	否	否	否	否
伤残?	否	否	否	是	否	否	否	否	否	否	否	否
体质指数	否	否	否	否	34	36	35	34	36	35	37	39
年龄	否	否	否	否	18	39	40	59	60	最大值	20	30
操作												
费用	1.5	1.5	1.5	1.5	1.75	1.75	1.5	1.5	1.25	1.25	2	2
拒绝	否	否	否	否	否	否	否	否	否	否	否	否

	测试用例											
条件	13	14	15	16	17	18	19	20	21	22	23	24
吸烟?	否	否	否	否	否	否	否	否	否	否	否	否
诊断?	否	否	否	否	否	否	否	否	否	否	否	否
住院?	否	否	否	否	否	否	否	否	否	否	否	否
伤残?	否	否	否	否	否	否	否	否	否	否	否	否
体质指数	38	37	39	38	16	40	0	最大值	17	33	20	30
年龄	45	55	65	75	35	50	25	70	37	47	0	17
操作												
费用	1.75	1.75	1.50	1.50	无	无	无	无	1	1	无	无
拒绝	否	否	否	否	是	是	是	是	否	否	是	是

	测试用例											
条件	25	26	27	28	29	30	31	32	33	34	35	36
吸烟?	是	否	否	否	是							
诊断?	否	是	否	否	是							
住院?	否	否	是	否	是							
伤残?	否	否	否	是	是							
体质指数	35	36	34	38	37							
年龄	20	50	70	30	35							
操作												
费用	2.625	2.25	1.875	3	10.125							
拒绝	否	否	否	否	否							

注意我所使用的测试非排他性规则的方法。首先，对于每一条规则，排他性的和非排他性的，我都独立地对它们进行了测试。然后，我对剩余的未测试的边界值进行了测试。接下来，我将一个非排他性规则和一个排他性规则组合在一起测试，确保每个非排他性规则至少进行过一次这样的组合测试（但对于所有的排他性规则，我们并不需要这么做）。最后，我将所有4条非排他性规则和一条排他性规则组合在一起测试。注意我并

未将非排他性和结果为“拒绝”的规则组合在一起进行测试，因为理论上我们就无法测试系统是否能正确计算费用了。

您可能注意到，我已经下意识地地对费用增长的最小值和最大值进行了测试。但是，我大概并未覆盖到所有可能的增长值。因为我并没有对这些规则组合所有可能的二元组、三元组、或者四元组进行测试，显然我也并未测试一个增长值所有可能的计算方法。我会在分类树和结对测试一节中对这个主题进行讲解。

在下表中我列出了对决策表和边界值的测试覆盖，以确保不会有所遗漏。下表中有3种覆盖度量用以显示对决策表的覆盖。

条件	1	2	3	4	5	6	7	8	9	10	11	12
吸烟?	是	—	—	—	—	—	—	—	—	—	—	—
诊断?	—	是	—	—	—	—	—	—	—	—	—	—
住院?	—	—	是	—	—	—	—	—	—	—	—	—
伤残?	—	—	—	是	—	—	—	—	—	—	—	—
体质指数	—	—	—	—	34～36	34～36	34～36	37～39	37～39	37～39	<17	>39
年龄	—	—	—	—	18～39	40～59	>59	18～39	40～59	>59	—	—
操作												
费用增长	50	50	50	50	75	50	25	100	75	50	—	—
拒绝	—	—	—	—	—	—	—	—	—	—	是	是
单个规则覆盖												
测试用例	1	2	3	4	5,6	7,8	9,10	11,12	13,14	15,16	17,19	18,20
成对规则覆盖												
测试用例	25	26	27	28	25	26	27	28				
所有规则组合覆盖												
测试用例	29	29	29	29				29				

下面是对体质指数的覆盖：

#	划　分	边 界 值	测 试 用 例
1	太瘦	0	19
2	太瘦	16	17
3	无增长	17	21
4	无增长	33	22
5	小幅增长	34	5
6	小幅增长	36	6
7	大幅增长	37	11
8	大幅增长	39	12
9	太重	40	18
10	太重	最大值	20

pin51.com 最新最全源代码 拼吾爱

下面是对年龄的覆盖：

#	划　分	边 界 值	测 试 用 例
1	太小	0	23
2	太小	17	24
3	年轻	18	5
4	年轻	39	6
5	中年	39	7
6	中年	59	8
7	老年	60	9
8	老年	最大值	10

4.2.18 用例

我们刚刚讲了决策表技术，这种技术在测试系统业务逻辑方面具有很强的优势。我刚才提到还有两种技术，用例和基于状态的测试技术，也能实现同样的目标。接下来，讲解用例测试技术。

> **ISTQB 术语**
> **用例测试**：一种黑盒测试设计技术，所设计的测试用例用于执行参与者场景。
> **用例**：参与者和系统进行对话过程中的一系列交互，能够产生实际的结果。

概念上而言，用例测试技术是用于通过不同视角设计测试，以确保测试到系统典型的和异常的工作流和场景的技术。通常要同时考虑从多种角度去设计测试用例，如从直接和系统进行交互的（各级）参与者的角度去考虑，以及从间接和系统交互的各种利益相关者的角度去考虑。如果我们（作为测试分析师）从业务分析师或系统设计师那里得到了用例，那么我们可以很方便地将这些用例作为我们设计测试用例的框架。

记住，在决策表测试中，我们关注的是交易情况，是指在某一指定时刻的单笔交易的一些条件，例如输入、前置条件等，足够触发系统做出相应的响应的情况。用例对这一点的要求没有这么严格，不过通常用例有一个假设，即典型的工作流是互相独立的。当典型的工作流无法执行时就会触发异常工作流，不过一般而言，前一个工作流和下一个工作流两者之间是独立的。至少对于正式的测试用例，我们可以通过明确地定义其前置条件和后置条件来确保这种独立性。前置条件和后置条件保证了在工作流的入口和出口某些事情为真。这样，我们就能将一个工作流和其他的工作流隔离开。此外，如果当前系统要处理的事件和条件与过去的事件和条件有很大的关联，那么我们可以使用基于状态的测试技术。

用例测试中使用的模型没有我们在等价类划分、边界值分析和决策表测试中使用的模型那么正式。实际上，“用例”这个词的概念在正式程度和表现形式上可以有很大的变化。用例的基本思想是用一系列编号标记的（或至少是线性的）步骤描述一个参与者如

何与系统交互。这些步骤可以用文本的形式来描述，也可以用流程图描述。

在用例中，经过一系列步骤之后会得到一个结果。这个结果通常应该对某些人来说是有价值的，比如直接与系统交互的参与者，或其他会间接受益于这个结果的利益相关者。

用例的步骤至少应该描述一个典型的工作流，也就是正常情况的处理过程。这个正常情况的处理过程有时候也被称为主场景、正常过程、基本过程、主要过程、正常流或恰当路径。然而，由于事情并不总是恰当的，因此在用例的这一系列步骤中也应该包含体现非正常处理过程，有时候也叫做异常、异常处理或替代过程。

比较正式的用例记录方法不仅要记录典型工作流和异常工作流，同时还要列出参与者、前置条件、后置条件、优先级、使用频率、特别需求、假设等。正式的记录方法可能也会要求创建一个用例图，在用例图中要列出所有的参与者、所有的用例以及参与者和用例之间的关系。

现在，我要做一个假设，实际上，这也是 ISTQB 大纲中暗含的假设，即您是从其他地方得到用例的，而不是直接创建的。如果去翻看 ISTQB 的高级大纲和基础级大纲，您会发现他们所谈到的用例是测试分析师从别人那里拿到的，然后测试分析师再根据这些拿到的用例设计测试用例。所以在此我并不会去尝试覆盖开发过程中所有可能的用例变体，而是选择其中一些基础的、非正式的用例和一些比较正式的用例用于测试设计，对于其他大量的用例变体，我这里就不再一一讲解了。

假设我们得到了一个用例，那么我们应该如何根据这个用例来设计测试用例呢？至少，我们可以先为每一个工作流设计一个测试用例，包括典型的和异常的工作流。如果要省略异常的工作流，那么您就应该指出为什么要这么做，您可以参考需求文档或其他相关文档。在进行用例测试的时候，未能对异常工作流进行测试是一个常见的错误。

根据用例设计测试的时候，您也可以使用等价类划分技术和边界值分析。事实上，当您发现存在条件组合决定操作的情况时，您可能发现了一个嵌入的、隐式的决策表。为了覆盖您在用例中发现的（等价类）划分、边界和业务规则，您可能最终要为每个工作流设计 2 个、5 个、10 个、20 个甚至更多的测试用例。

记住，我说过用例应该有一个明确的结果。评估测试结果的过程就包含了对用例结果的验证，而不仅仅是对在测试该工作流过程中得到的恰当的界面、消息、输入确认进行验证。

注意此处隐含的一个覆盖准则：每一个工作流，典型的和异常的，都至少要有一个测试用例。这并不是一个正式的准则，不过非常有效。

那么在用例测试中我们的缺陷假设是什么呢？我们在决策表测试中查找的是会导致系统执行错误操作或导致系统不执行正确操作的条件组合。而对于用例，我们的假设就比较粗略了。这里，我们查找的是系统无法与顾客正确交互或返回不正确结果的情况。

图 4-32 给出了一个非正式用例的示例，该用例描述了从电子商务网站购物的过程，比如像我们在决策表中提到过的 rbcs-us.com 网站。

在图的上方是网站购物的正常工作流。恰当路径如下：

1. 顾客往购物车内放入至少一项商品。

pin51.com 最新最全源码资源下载 拼吾爱

电子商务购物：正常工作流

1. 顾客往购物车内放入至少一项商品。
2. 顾客选择结账。
3. 系统从顾客那里收集地址、支付方式和送货方式等相关信息。
4. 系统显示所有信息并要求顾客确认。
5. 顾客确认订单。

异常

- 在购物车为空的情况下顾客试图结账；系统返回错误信息。
- 顾客提供的地址、支付方式或送货方式等信息无效；系统返回适当的错误信息。
- 顾客在结账之前或结账过程中放弃交易；若顾客超过 10 分钟没有下一步动作，那么系统将登出顾客。

图 4-32　非正式用例示例

2. 顾客选择结账。
3. 系统从顾客那里收集地址、支付方式和送货方式等相关信息。
4. 系统显示所有信息并要求顾客确认。
5. 顾客确认订单。

注意上述工作流的最终结果是系统收到订单，并准备发货。所以应该有与完成订单相关的用例，在这个用例中将描述从系统收到订单、到货物送达顾客家里或工作场所的这个过程。

我们再来看看购物用例的异常工作流：

- 首先，顾客试图在购物车为空时结账。在这种情况下，系统返回错误信息。
- 另外，顾客提供的地址、支付方式和送货方式等信息可能是无效的。在典型的电子商务工作流中，如果我们遇到上述信息无效的情况，那么系统会以适当的形式给出错误消息，并停止进一步处理，直到顾客解决错误。
- 最后，顾客可能在结账之前或结账过程中放弃交易；若顾客超过 10 分钟不活动，那么系统将登出顾客。

现在，我们将为这个用例设计测试用例。在图 4-33 中，可以看到用于覆盖典型工作流的测试规程的主体部分。（为了更简洁明了，我去掉了典型的测试规程中应有的头部和尾部信息。）

正如您所看到的，工作流中的每一步都对应着测试规程中的一个步骤。同时，我还对商品数量、支付类型和送货地址进行了等价类划分和边界值分析。因为所有的信息都是有效的，所以我将它们组合在一起。此处我使用的是一种节省空间的方式，描述应该如何在测试规程各个变体中重复核心测试步骤（1～5 项），而不是在该表下部的（6～8 项）测试规程中再完整地重复这些核心测试步骤。

在图 4-34 中您可以看到覆盖异常工作流的测试规程的主体部分。同时，我还对顾客放弃交易的时间点进行了等价类划分。

此处我们正好谈谈逻辑测试用例和具体测试用例之间的重要区别。若是为了通过 ISTQB 的考试，那么您要知道这两个词的术语定义。不过对于本书的读者而言，我们只要知道逻辑/概要测试用例描述了测试条件和结果。而具体/详细测试用例还要求列出能满

pin51.com 新客发

	测 试 步 骤	预 期 结 果
1	往购物车内放入一项商品	显示购物车内商品
2	单击结账按钮	系统显示结账界面
3	输入有效的美国地址、用 American Express 有效支付、以及发货信息	每一个界面都能正确显示、并接受正确的输入
4	检查订单信息	显示信息与用户输入一致
5	确认订单	系统接受订单
6	重复步骤 1～5，但是这次往购物车中添加 2 项商品，用 Visa 支付，并发往其他国家	如 1～5 步中的预期结果所示
7	重复步骤 1～5，但是这次往购物车中添加最大数量的商品并用 Mastercard 支付	如 1～5 步中的预期结果所示
8	重复步骤 1～5，但这次用 Discover 支付	如 1～5 步中的预期结果所示

图 4-33 设计测试用例示例（典型的）

足测试条件的输入数据和测试结果中观察到的输出数据。

	测 试 步 骤	预 期 结 果
1	不往购物车内放置任何商品	购物车是空的
2	单击结账按钮	显示错误信息
3	往购物车内放入商品，单击结账按钮，输入无效的地址、无效的支付信息，以及无效的发货信息	显示错误信息，除非解决问题，否则不能进入下一步
4	检查订单信息	显示信息与用户输入一致
5	确认订单	系统接受订单
6	重复 1～3 步，但在往购物车内放入商品后放弃交易，并停止操作	用户在最后一次操作之后的 10 分钟准时登出
7	重复 1～3 步，然后在每个界面上放弃交易并停止操作	如第 6 步的预期结果
8	重复 1-4 步，但不要确认订单	如第 6 步的预期结果

图 4-34 设计测试用例示例（异常的）

ISTQB 术语

概要（或逻辑）测试用例：没有具体的（实现级别）输入数据和预期结果的测试用例。实际值没有定义或是可变的，而用逻辑概念来代替。

详细（或具体）测试用例：具有具体的（实现级别）输入数据和预期结果的测试用例。抽象测试用例中所使用的逻辑运算符被替换为对应于逻辑运算符作用的实际值。

正如在图 4-33 和图 4-34 中看到的，您可以很容易地将一个用例转换成一个或多个逻辑测试用例。不过，若要将逻辑测试用例转换成具体测试用例，那么您可能需要其他的文档。例如，我们可以在购物车中最多放多少商品？为了得到这个确切的数字，我们

pin51.com 拼吾爱 最新最全源码分享

需要参考其他文档，理想情况下应该是一个需求规格说明。又例如什么样的商品可以放入购物车？这时候就需要一些对商店库存的描述信息。

我们定义了逻辑测试用例而不是具体测试用例，这种做法是不是在作弊？不，当然不是。因为无论怎样，在某个时间点，一个测试用例必须变为具体的。您必须输入明确的输入值，同时您也必须验证明确的输出值。在ISTQB基本测试过程中，从逻辑测试用例到具体测试用例的转换被认为是一种实现活动。如果您决定将测试用例的实现交由测试人员在测试执行过程中进行，没问题，不过您必须确保测试执行的时候有足够的信息去实现测试用例。不然，您就是在冒风险，因为这可能导致测试进度延迟。

那么，在正式的用例中有什么不同呢，或者说正式的用例中多了哪些信息呢？通常，一个正式的用例比非正式的用例要包含更多的信息。这里，您可以看到一个正式用例中应该包含的一些典型的元素：

- ID：用例识别号。
- 名字：简称，比如电子商务购物。
- 参与者：比如顾客。
- 描述：用例的简要描述。
- 优先级：从实现的角度而言。
- 使用频率：用例发生的频率。
- 前置条件：在正常执行用例之前需要满足哪些条件。
- 典型工作流：通常情况下会使用我们之前介绍非正式用例时使用的描述形式，但有时候也会分成两列，一列描述参与者的操作，一列描述系统响应。
- 异常工作流：每一个异常都有一个工作流，通常还会有专门的栏描述参与者的操作以及系统的响应。
- 后置条件：在用例正常完成之后，系统的状态应该满足什么条件。

作为一个测试分析师，您可以使用其中的一部分信息。而有一些信息，例如优先级和使用频率，除非是在进行风险分析，否则您是用不到的。还有，要注意对工作流进行细分，尤其是异常工作流，工作流的粒度越小（或划分得越精细），那么您的测试就会有更高的可追溯性。

图4-35中是我们曾讲到过的非正式用例的正式版本的信息头。注意，非正式用例中

ID	02.001
用例名	电子商务购物
参与者	顾客
描述	顾客能够往购物车内放入商品并完成交易
优先级	非常高
使用频率	每天最多有1 000个顾客，其中25%的顾客会用到这项功能
前置条件	1. 购物车内有一项或多项商品 2. 顾客已经登录 3. 顾客单击结账按钮

图4-35 正式用例示例（第1部分）

的一些步骤在这里变成了前置条件。这意味着挑选商品的过程被独立出来，变成了一个单独的用例，而本用例将完全专注于描述电子商务网站的购物结账功能。还有，我们以前也不知道“登录”相关的需求。而对于我们所进行的测试而言，这也是非常重要的信息。

图 4-36 中是该正式用例的主体部分，包括一个正式的工作流和 3 个异常工作流。其中，正常工作流比以前短了，因为其中的一些步骤变成了前置条件。在正常工作流的下方是异常工作流，每一个异常占了一行。最后，只有当正常工作流全部完成之后，后置条件才会为真。

典型工作流	1. 系统从顾客那里收集地址、支付以及发货等相关信息 2. 系统显示所有信息并要求顾客确认 3. 顾客确认订单并等待发货
异常工作流 1	购物车为空时，顾客试图结账 系统显示错误信息
异常工作流 2	顾客提供无效的地址、支付或发货信息 系统显示恰当的错误信息
异常工作流 3	顾客在结账前或在结账过程中放弃交易 系统将超过 10 分钟无操作的顾客登出
后置条件	订单在系统中处于待处理状态

图 4-36　正式用例示例（第 2 部分）

4.2.19　用例练习

参考 HELLOCARMS 系统需求文档的 001 节：一个非正式用例。现在假设在这个非正式用例的第 5 步中，系统会提供我们之前练习中提到的人寿保险。

该练习包含两步：

1. 将 001 节中的非正式用例改写成比较正式的版本，要求列出参与者、前置条件、异常工作流、后置条件。同时记得要增加保险功能。

2. 以测试规程的形式设计逻辑测试用例用以覆盖改写后的用例，同时适当地使用等价类划分技术和边界值分析技术。

这两个部分的答案在下面。在您查看第二部分的答案前，应该先查看第一部分的答案，以和自己的答案对照（并且，有必要的话，修改自己第二部分的答案）。

4.2.20　用例练习解答

为了方便阅读，此处列出了 001 节中的非正式用例。

下面的这个非正式用例描述了 HELLOCARMS 系统中的典型业务：

1. Globobank 客服中心的一位电话银行经理接到了一个顾客打来的电话。

2. 电话银行经理对顾客进行了面试，并使用他们桌面上的网页浏览器打开 HELLOCARMS 系统，输入相关信息。如果顾客要求贷款的数目比较大，或者他们房产

pin51.com 最新最全源码分享 拼吾爱

价值比较大，那么电话银行经理会将这个申请提交给高级电话银行经理，因为高级电话银行经理可以决定如何处理这个申请。

3. 一旦电话银行经理从顾客那里收集到信息以后，HELLOCARMS 系统会根据评分主机中返回的信息确定顾客的信用等级。

4. 根据所有已有的顾客信息，如果有的话，HELLOCARMS 系统会列出电话银行经理可以向顾客提供的各种相应的房屋净值（金融）产品。

5. 如果顾客选择了上述产品中的一个，那么电话银行经理会有条件地确认这个申请。

6. 在面试结束之后，电话银行经理会指示 HELLOCARMS 系统将贷款信息移交给位于洛杉矶数据中心的贷款文件打印系统（LoDoPS），打印初始的贷款文件。

7. 当下列事件发生时，HELLOCARMS 系统会收到一份来自于 LoDoPS 系统的更新信息：

a. LoDoPS 系统向顾客发送了贷款文件。

b. Globobank 的贷款服务中心收到了顾客签名确认的贷款文件。

c. Globobank 的贷款服务中心向顾客寄出了支票以及其他与顾客所选产品相关的材料。

注意，其实第 7 步属于另一个用例，因为在第 6 步的结尾，用例已经得到了一个有用的结果，即那个被提交给系统的申请。

因此，如表 4-3 所示，只是将第 1 步到第 6 步改写成了半正式的形式。另外我在第 5 步（异常工作流 3）中也提到了人寿保险功能。

表 4-3 半正式的 HELLOCARMS 用例

参与者	电话银行经理
前置条件	Globobank 的电话银行经理已经登录 HELLOCARMS 系统
正常工作流	1. Globobank 客服中心的一位电话银行经理接到了一个顾客打来的电话 2. 电话银行经理对顾客进行了面试，并使用他们桌面上的网页浏览器打开 HELLOCARMS 系统，输入相关信息 3. 一旦电话银行经理从顾客那里收集到信息以后，HELLOCARMS 系统会根据评分主机中返回的信息确定顾客的信用等级 4. 根据所有已有的顾客信息，如果有的话，HELLOCARMS 系统会列出电话银行经理可以向顾客提供的各种相应的房屋净值（金融）产品 5. 如果顾客选择了上述产品中的一个，那么电话银行经理会有条件地确认这个申请 6. 在面试结束之后，电话银行经理会指示 HELLOCARMS 系统将贷款信息移交给位于洛杉矶数据中心的贷款文件打印系统（LoDoPS），等待打印初始的贷款文件
异常工作流 1	在正常工作流的第 2 步，如果顾客要求贷款的数目比较大或者他们房产价值比较大，那么电话银行经理会将这个申请提交给高级电话银行经理，因为高级电话银行经理可以决定如何处理这个申请 如果高级经理决定接受申请，那么电话银行经理会完成第 2 步的剩余工作，并继续执行正常工作流 如果高级经理决定拒绝申请，那么电话银行经理会通知顾客他的申请已经被拒绝，并结束面试

续表

异常工作流 2	在正常工作流的第 4 步，如果系统没有列出任何可提供给顾客的房屋净值产品，那么电话银行经理会通知顾客他的申请已经被拒绝，并结束面试
异常工作流 3	在正常工作流的第 5 步，如果顾客选择了房屋净值贷款，那么电话银行经理将会询问顾客是否要购买一个人寿保险，以便在发生意外时偿还贷款。如果顾客表示愿意购买，那么接下去的步骤如下： 1. 电话银行经理对顾客进行了面试，并使用他们桌面上的网页浏览器打开 HELLOCARMS 系统，输入健康相关的信息 2. HELLOCARMS 系统会处理我们在上一个练习中提到的保险相关信息，并得到下面这两个结果中的一个结果： a. 根据顾客提供的健康信息，HELLOCARMS 系统决定拒绝提供保险。电话银行经理告诉顾客他的保险申请被拒绝了。异常工作流结束并返回第 5 步 b. HELLOCARMS 系统根据顾客提供的健康信息以及其贷款额度计算出保险费用。电话银行经理告知顾客 3. 顾客决定是否接受这个报价并购买保险： a. 接受报价。电话银行经理将人寿保险交易作为贷款申请的一部分处理。异常工作流结束，并返回第 5 步 b. 拒绝报价。电话银行经理将人寿保险交易排除在贷款申请之外，并返回第 5 步
异常工作流 4	在正常工作流的第 1 步到第 5 步的过程中，如果顾客决定结束面试，不再继续申请或选择产品，那么贷款申请过程取消，面试结束
异常工作流 5	如果在电话银行经理未登录系统（例如，因为系统未启动）的情况下，遇到正常工作流的第 1 步，那么接下去的步骤如下： 1. 电话银行经理继续手动地记录信息。在面试结束的时候，电话银行经理会告诉顾客：他会稍晚的时候回电，告知是否接受申请 2. 一旦电话银行经理登录系统之后，他会将顾客的申请信息输入到 HELLOCARMS 系统中，并重新开始正常工作流的第 2 步 3. 在下列情况中电话银行经理会给顾客打电话： a. 遇到正常工作流的第 5 步。从第 5 开始继续处理申请 b. 在正常工作流的第 2 步，触发了异常工作流 1。从第 2 步开始继续处理申请 c. 在正常工作流的第 4 步，触发了异常工作流 2。不再继续申请流程
后置条件	贷款申请文件已被移交给 LoDoPS 系统，等待打印初始的贷款文件

现在，我将以测试规程的形式设计逻辑测试用例用以覆盖这个用例。在这里，将使用我曾介绍过的模式，将测试规程分成正常工作流规程（见表 4-4）以及为每个异常工作流设计一个异常规程（见表 4-5、表 4-6、表 4-7、表 4-8 和表 4-9）。

表 4-4 正常工作流测试规程

#	测试步骤	预期结果
1	在 HELLOCARMS 系统中为一个信用良好的顾客启动房屋净值贷款的申请流程，填写相关信息	确认界面和消息
2	将顾客信息提交给 HELLOCARMS 系统，并通过评分主机确定顾客信用	验证与评分主机的交互
3	检查可向顾客提供的各种房屋净值产品	验证系统列出的产品是恰当的 确认界面和消息

pin51.com 最新最全软件资料分享 拼吾爱

续表

#	测试步骤	预期结果
4	选择房屋净值贷款	确认界面和消息
5	命令 HELLOCARMS 系统将贷款信息移交给贷款文件打印系统（LoDoPS）	验证与 LoDoPS 系统的交互
6	重复 1～5 步，不过这次在第 4 步中选择房屋净值信贷额度	如 1～5 步所示
7	重复 1～5 步，不过这次在第 4 步中选择反向住房按揭贷款	如 1～5 步所示

表 4-5 异常工作流 1 的测试规程

#	测试步骤	预期结果
1	在 HELLOCARMS 系统中为一个信用良好的顾客启动房屋净值贷款的申请流程，填写相关信息	确认界面和消息
2	输入的贷款申请数目超过了最大值	验证申请移交界面
3	输入“继续”命令的授权代码	验证申请流程会在正常工作流的第 2 步重新开始
4	完成正常工作流	验证贷款申请已被正确提交给 LoDoPS 系统
5	重复 1～2 步，但这次输入一个超过最大值的房产价值，并在第 3 步中输入“拒绝”命令的授权代码	验证显示的“拒绝”脚本并且申请流程结束
6	重复 1～4 步，但这次输入的贷款申请数目和房产价值都超过了它们相应的最大值。	如 1～4 步所示

表 4-6 异常工作流 2 的测试规程

#	测试步骤	预期结果
1	在 HELLOCARMS 系统中为一个信用不好的顾客启动房屋净值贷款的申请流程，填写相关信息	确认界面和消息
2	将顾客信息提交给 HELLOCARMS 系统，并通过评分主机确定顾客信用	验证与评分主机的交互
3	检查可向顾客提供的各种房屋净值产品	验证系统未列出任何产品 确认界面和消息 验证显示的“拒绝”脚本并且申请流程结束

表 4-7 异常工作流 3 的测试规程

#	测试步骤	预期结果
1	在 HELLOCARMS 系统中为一个信用良好并且具备保险资格的顾客启动房屋净值贷款的申请流程，填写相关信息	确认界面和消息
2	将顾客信息提交给 HELLOCARMS 系统，并通过评分主机确定顾客信用	验证与评分主机的交互

续表

#	测 试 步 骤	预 期 结 果
3	检查可向顾客提供的各种房屋净值产品。并选择房屋净值贷款。同时顾客愿意申请保险	验证系统提供的产品是恰当的 确认界面和消息
4	输入顾客的健康信息	验证界面和消息
5	检查系统提供的保险，接受保险费用报价和完成贷款申请，包括将申请信息发送至 LoDoPS 系统	验证保险费用。 验证人寿保险的申请信息与贷款申请信息都被移交给 LoDoPS 系统
6	重复 1～5 步，不过这次拒绝报价	如 1～5 步所示，但这次验证人寿保险申请信息未被移交给 LoDoPS 系统
7	重复 1～5 步，不过这次的顾客不具备保险资格	如 1～5 步所示，但这次系统没有提供任何人寿保险并且也没有人寿保险申请信息被移交给 LoDoPS 系统

表 4-8 异常工作流 4 的测试规程

#	测 试 步 骤	预 期 结 果
1	在 HELLOCARMS 系统中为一个信用良好的顾客启动房屋净值贷款的申请流程，填写相关信息	确认界面和消息
2	将顾客信息提交给 HELLOCARMS 系统，并通过评分主机确定顾客信用	验证与评分主机的交互
3	检查可向顾客提供的各种房屋净值产品	验证系统列出的产品是恰当的 确认界面和消息
4	表示顾客要取消申请	确认显示的界面 验证贷款申请已经取消
5	重复 1～2 步，但在倒数第二个界面取消申请	如第 4 步所示
6	重复第 5 步，在每个可能的界面上取消申请，包括第一个界面	如 1～5 步所示
7	重复 1～5 步，但在第 4 步中选择房屋净值信贷额度	如 1～5 步所示

表 4-9 异常工作流 5 的测试规程

#	测 试 步 骤	预 期 结 果
1	启动手动申请输入模式。 在 HELLOCARMS 系统中启动房屋净值贷款的申请流程（贷款数量和房产价值正常，且顾客信用良好）	确认界面和消息 验证应该在正常工作流的第 5 步给顾客打电话 验证从那时起能正常处理并提交信息
2	重复第 1 步，不过这次申请的贷款额度超过其最大值。当遇到请求的时候输入“继续”命令的授权代码	确认界面和消息 验证应该在正常工作流的第 2 步给顾客打电话 验证从那时起能正常处理并提交信息
3	重复第 1 步，不过这次的顾客信用不好。	确认界面和消息 验证应该在正常工作流的第 4 步给顾客打电话 验证申请已被取消

pin51.com 最新最全源码分享下载 拼吾爱

我对顾客进行了等价类划分如下：

- 信用良好
- 信用不好

您也可以在这里使用边界值分析技术，根据顾客的信用评分，测试那些信用刚够申请贷款的顾客和还差一点才能申请贷款的顾客。

我对贷款产品的等价类划分如下：

- 房屋净值贷款
- 房屋净值信贷额度
- 反向住房按揭贷款

同样，在这里您可以对上面每一个贷款类别进行边界值分析，申请最小的贷款额度和最大的贷款额度。

我还对向高级银行经理移交申请的情况进行了划分，得出的等价类如下：

- 房产价格太高
- 贷款额度太大

我建议还应该对我们曾在本节第二个练习中得出的边界值和等价类进行测试。

4.2.21 基于状态的测试与状态转换图

前述章节中，我曾提及在我们讨论过等价类划分技术和边界值分析技术之后，我们还将讨论3种技术。这3种技术比起等价类划分技术和边界值分析技术，更适合用于测试业务逻辑。然后，首先讲了决策表，这种技术对于业务场景测试最为有效。再是用例测试技术，用例的前置条件和后置条件能将一个工作流与前一个工作流和后一个工作流隔离开。

现在，让我们来讲讲基于状态的测试技术。当有一系列的事件和条件，且对特定事件/条件的处理又取决于曾经发生过的事件和条件时，使用基于状态的测试技术就比较理想。在某些情况中，事件的序列可能是无限的，这显然超过了我们的测试能力，但我们想要有这样的一种测试设计技术，使得我们能够测试任意长度的事件序列。

基于状态的测试技术的基本模型是一个状态转换图或状态转换表。在图或表格中，起始的状态、事件和条件与结果状态和操作连接在一起。

也就是说，系统经历过某些状态之后，到达了当前状态。然后，发生了某个系统必须响应的事件。系统对该事件的处理可能会受到一个或多个条件的影响。这种事件/条件组合触发了一次状态转换，从当前状态转换到一个新的状态或再次回到当前状态。在状态转换的过程中，系统执行了一个或更多个操作。

得到这个模型之后，我们可以设计测试用例去遍历这些状态和转换过程。输入触发了事件并创造了条件，而测试的预期结果是新的状态和系统执行的操作。

在基于状态的测试中可以使用各种覆盖准则。最弱的准则要求测试遍历每一个状态以及每一次转换。我们可以在状态转换图中使用这个准则。一个更高级别的覆盖准则是：状态转换表中的每一行至少要有一个测试用例。在您覆盖了表格中的“每行”之后，那么也必定覆盖了“每一个状态和每一次转换”，这就是为什么我称其为更高级别的覆盖准

pin51.com 拼五网 提供海量分享资源

则的原因。

ISTQB 术语

状态转换测试：一种黑盒测试设计技术，所设计的测试用例用来执行有效和无效的状态转换。

另一个可能更高级别的覆盖准则要求长度小于等于 N 的每一条状态序列至少有一个对应的测试用例。N 可以是 1、2、3、4 或者更高。这种覆盖准则有时被称为周氏切换覆盖（Chow's switch coverage），以其发明者周教授的姓氏命名，或者也叫做 N-1 切换覆盖，根据给定的覆盖程度/级别命名。如果您要覆盖所有长度为 1 的转换，那么 N-1 切换覆盖意味着 0-切换覆盖。这就跟我们刚才讲到的最低级别的覆盖是一样的。如果您要覆盖所有长度为 1 和 2 的转换，那么 N-1 切换覆盖意味着 1-切换覆盖。当然，这种覆盖准则比最低级别的覆盖准则要高一些。

1-切换覆盖并不一定比“每行”覆盖更高级。这是因为在状态转换表中的某些状态和事件/条件的组合并不会出现在状态转换图中。N-1 切换覆盖中所谓的“切换”来自于状态转换图，而非状态转换表。

如果您对基础级中测试设计的内容已经有点模糊了，那么可能会觉得我们上面讲的那些覆盖准则难以理解。对于那些为了准备基础级考试而填鸭式学习的人来说这种问题很常见。不过，不要着急，您很快就会明白了。

那么，基于状态的测试是基于什么样的缺陷假设呢？我们希望找到：根据到目前为止的事件/条件组合的历史，系统在一组条件下，对特定事件做出了错误的响应，比如执行了错误的操作或转换到了错误的状态。

图 4-37 中是电子商务网站上购物过程的状态转换图。该图从顾客的角度描述了系统与顾客的交互。现在让我们来仔细看看这幅图，在这个过程中，我会指出通常一个状态转换图应有哪些关键元素，以及本图又有什么特点。

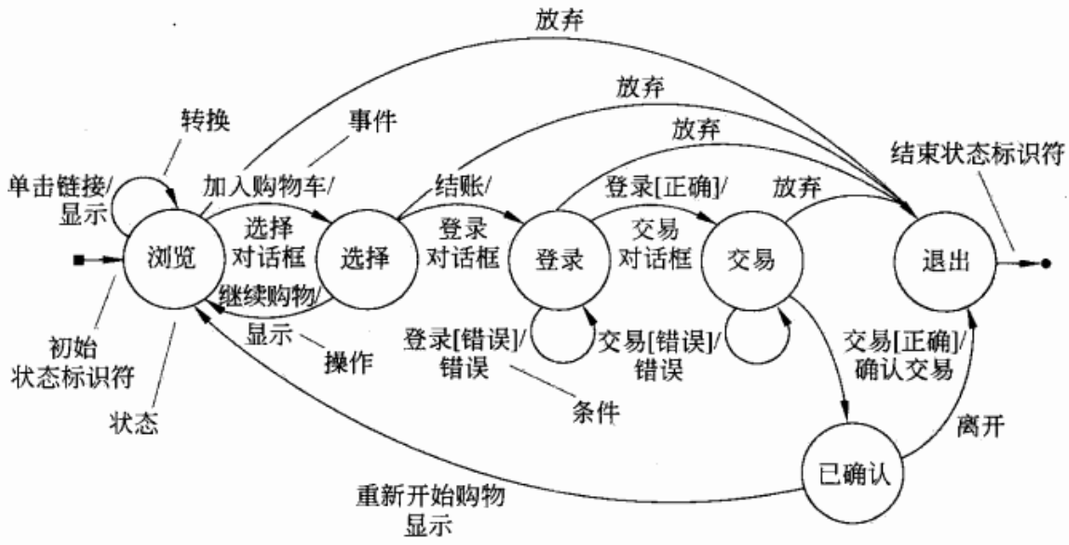


图 4-37 状态转换图示例

pin51.com 最新最全软件资源 拼吾爱

首先，可以看到，在图的最左端有一个很小的名为“初始状态标识符”的点-箭头元素。这个符号表示，从顾客的角度看，当他们开始浏览网站的时候业务就开始了。在“浏览”状态中，我们可以单击商品的链接和浏览商品目录。注意“浏览”状态上方的循环箭头。如“浏览”状态下方的标识所示，节点或泡泡代表状态。如循环箭头上方的标识所示，箭头代表转换。

接下来，我们向购物车中添加一个商品之后就进入了“选择”状态。如箭头上方的标识所示，“添加到购物车”是一个**事件**（event）。在将商品添加到购物车的过程中，系统会显示一个“选择对话框”，要求顾客告知想要购买的商品数目和其他所需信息。完成这一步之后，顾客可以告诉系统他们想要继续购物，在这种情况下，系统会再次显示主界面，即回到“浏览”状态。从图中看，在带箭头的线段上，并位于**事件**以及一个斜杠符号下方的标识代表系统执行的**操作**（action）。

或者，顾客在“选择”状态中决定结账。这时候，他们进入“登录”状态并输入登录信息。对于登录信息有一个条件：信息正确或信息错误。如果信息错误，那么系统返回一条错误消息，而顾客依旧停留在“登录”状态。如果信息正确，那么系统将显示交易对话框的第一个界面。注意方括号中的“正确”和“错误”就是**条件**（condition）的符号表示。

在“交易”状态中，系统会要求顾客输入支付信息。支付信息有正确和错误之分，并决定了我们是否能够完成这笔交易。一旦顾客确认了交易，那么他可以继续购物或离开。

在培训课程中，当我们讲到基于状态的测试时，学员们经常会问“我应该怎么去区分状态、事件和操作这三者？”主要的区别如下：

- 一个状态可以持续存在，直到系统之外的某个事件触发了一次转换。一个状态可以不限期地存在下去。
- 事件在是瞬间发生的或者只会持续有限的时间。事件是指发生在系统外部的，能触发转换的某些事物。
- 操作是指系统在状态转换期间的响应。操作与事件类似，是瞬间发生的或只会持续有限的时间。

也就是说，有时候可以用不同的方式去描述同一种情况，尤其当某一个状态或操作可以被进一步细分成更小粒度的状态、事件和操作的时候。我们稍后来看这样的一个例子，将“交易”状态细分成各个子状态。

最后要注意的是，在一开始的时候，我说过我是从顾客的角度去画图 4-37 的。如果换成从系统的角度去画图，那么结果将有所不同。在画这些图形的时候，关键的一点就是要保持同一个视角；不然，就会发生一些没有意义的事情。

基于状态的测试技术使用的是正式的模型，因此我们可以使用一个正式规程来根据这个模型设计测试用例。下面列出的这个规程可以用于设计测试用例，并能满足（基本）状态/转换覆盖（即，0 切换覆盖）。

1. 要明确一个规则，即测试规程或测试步骤必须在哪里开始以及“可以”或“必须”在哪里结束。例如，一个测试步骤必须从初始状态开始，然后只能在最终状态结束。我

们之所以使用了“可以”或“必须”这两个词语是因为有可能起始状态和最终状态是同一个，因此各种状态和转换的序列有可能会不止一次地经历初始状态。

2. 从一个有效的测试起始状态开始，定义一系列事件/条件组合，直到到达一个有效的测试结束状态。对于其中的每一次转换，列出系统应该执行的预期操作。这就是测试的预期结果。

3. 当您每遍历过一个状态和转换之后，就将它们标识为已覆盖的。有一种简单的方法，就是将状态转换图打印出来之后，在您覆盖过的节点和箭头上打一个标记。

4. 重复第 2 步和第 3 步，直到您已经遍历过所有的状态和转换，换句话说，在您的图上，您已经标记过了所有的节点和箭头。

可以使用上述规程来设计逻辑测试用例。但如果要设计具体测试用例，那么就必须给出实际的输入值和实际的输出值。因此，我在介绍这种技术的时候使用的是逻辑测试用例，不过记住，正如我曾提到过的，在测试执行之前您必须实现这些具体测试用例。

现在我们将使用这个过程为刚讨论过的电子商务应用程序设计测试用例。如图 4-38 所示，虚线代表覆盖到的状态和转换。这里，有两件事情：

- 首先，我们规定测试必须在初始状态开始，并在最终状态结束。
- 接下来，我们设计了第一个测试步骤（浏览、单击链接、显示、加入购物车、选择对话框、继续购物、显示、加入购物车、选择对话框、结账、登录对话框、登录[错误]、错误、登录[正确]、交易对话框、交易[错误]、错误、交易[正确]、确认、继续购物、显示、放弃、退出）。

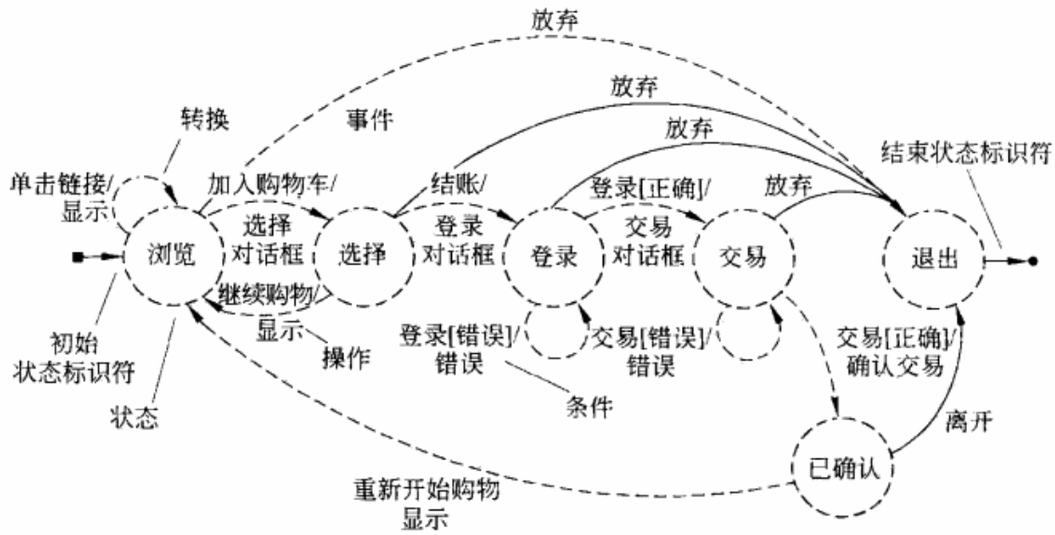


图 4-38 覆盖检查 1

现在，在我们的状态转换图中检查一下覆盖的完成程度。

正如您所看到的，我们覆盖了所有的状态以及大部分的转换，但还是有一些转换未覆盖到。因此我们要设计更多的测试用例去覆盖这些转换。

图 4-39 显示的是下列额外的测试步骤所满足的测试覆盖：

1.（浏览、单击链接、显示、加入购物车、选择对话框、继续购物、显示、加入购

pin51.com 最新最全源码分享 拼吾爱

物车、选择对话框、结账、登录对话框、登录[错误]、错误、登录[正确]、交易对话框、交易[错误]、错误、交易[正确]、确认、继续购物、显示、放弃、退出）。

2.（浏览、加入购物车、选择对话框、放弃、<无操作>、退出）。

3.（浏览、加入购物车、选择对话框、结账、登录对话框、放弃、<无操作>、退出）。

4.（浏览、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。

5.（浏览、加入购物车、选择对话框、继续购物、显示、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、交易[正确]、确认、离开、<无操作>、退出）。

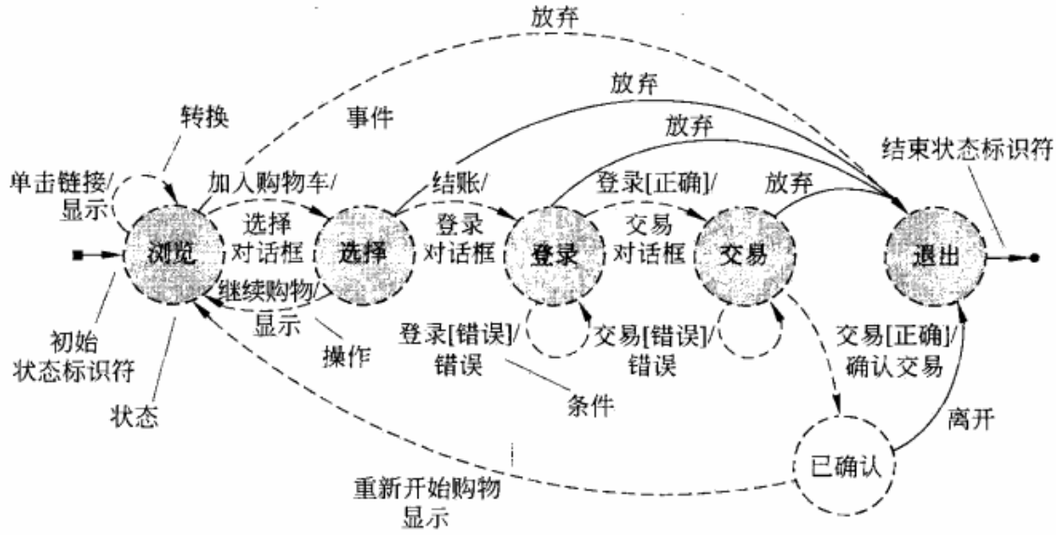


图 4-39 覆盖检查完成

图 4-39 显示了对状态和转换的覆盖。如果您还未标记每一个状态和每一个转换，那么就应该设计更多的测试用例。

4.2.22 状态和子状态

在某些情况中，将一个状态视为超级状态，并将它细分为两个或多个子状态是有意义的。在图 4-40 中，可以看到我将电子商务这个例子中的交易状态分为 3 个子状态。

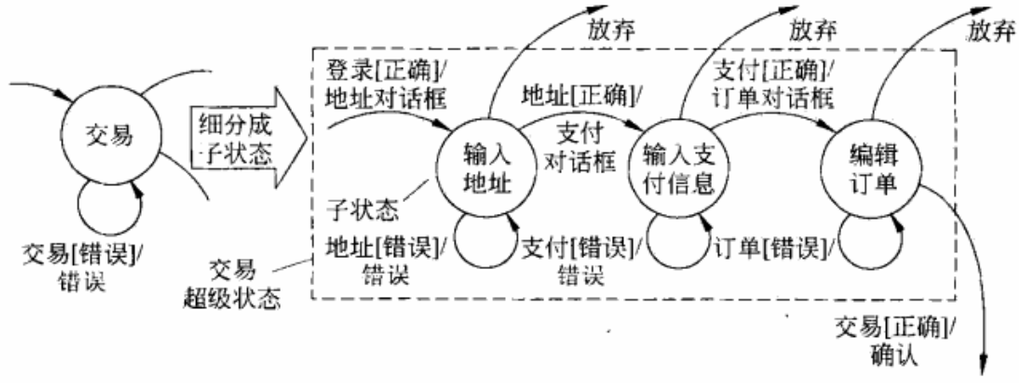


图 4-40 超级状态和子状态

这里，基本覆盖的规则很简单。即覆盖所有进入超级状态的转换、所有离开超级状态的转换、所有子状态以及超级状态内的所有其他转换。

注意，在我们的例子中，将超级状态细分成子状态会增加测试用例的数目，因为原先从“交易”这个超级状态到“离开”这个状态的“放弃”转换现在变成了 3 个。这意味着我们的测试中多了一个细粒度的元素，例如，有更多事件和操作。同时我们要确保至少对 3 种类型的错误交易输入进行了测试。

4.2.23 状态转换表

状态转换表很有用，因为它们强迫我们，以及业务分析师和系统设计师，去考虑那些我们可能遗忘的状态和事件/条件的组合。

若要创建一个状态转换表，首先要列出状态转换图中的所有状态。接下来，列出状态转换图中所有的事件/条件组合。然后，创建一个表格，对于每一个状态，将它与每一个事件/条件组合一起填入表中。表中的每一行有 4 个域：

- 当前状态。
- 事件/条件。
- 操作。
- 新状态。

对于表格中的某些行，在状态转换图中根据给定的当前状态和事件/条件的组合，我们可以找到相应的操作和新状态，并填入表格中。而对于表格中的其他行，这两个域的值是未定义的。

现在，我们可以问业务分析师、系统设计员和其他人，“那么，这些未定义的情况到底会发生什么呢？”

他们可能会说“喔，不可能发生这种情况！”作为一个测试分析师，您知道这意味着什么？您现在的工作是要想办法去触发这些未定义的情况。

他们可能会说“喔，嗯，我没有考虑到这种情况。”如果现在还是系统设计阶段，那么这意味着您可能找到了一个潜在的缺陷。

图 4-41 是电子商务程序的状态转换表的一部分。图中有 6 个状态：

- 浏览。
- 选择。
- 登录。
- 交易。
- 确认。
- 退出。

有 11 个事件/条件组合：

- 单击链接。
- 加入购物车。
- 继续购物。
- 结账。

pin51.com 最新最全源码分享下载 拼吾爱

				当前状态	事件/条件	操作	新状态
				浏览	单击链接	显示	浏览
		单击链接		浏览	加入购物车	选择对话框	选择
		加入购物车		浏览	继续购物	未定义	未定义
		继续购物		浏览	结账	未定义	未定义
浏览		结账		浏览	登录[错误]	未定义	未定义
选择		登录[错误]		浏览	登录[正确]	未定义	未定义
登录	×	登录[正确]	=	浏览	交易[错误]	未定义	未定义
交易		交易[错误]		浏览	交易[正确]	未定义	未定义
确认		交易[正确]		浏览	放弃	<无操作>	退出
退出		放弃		浏览	重新开始购物	未定义	未定义
		重新开始购物		浏览	离开	未定义	未定义
		离开		选择	单击链接	未定义	未定义
				（还有其他的53行未列出，生成方式与上述模式相同）			
				退出	重新开始购物	未定义	未定义

图 4-41 状态转换图示例

- 登录[错误]。
- 登录[正确]。
- 交易[错误]。
- 交易[正确]。
- 放弃。
- 重新开始购物。
- 离开。

这意味这个完整的状态转换表有66行，每一对特定状态与事件/条件组合就是一行。

要设计测试用例覆盖状态转换表，可以采用下列步骤。注意，我们刚才在讲解状态转换图的时候设计了满足（基本）状态/转换覆盖或0切换覆盖的测试用例，现在我们可以重用这些测试用例。

1. 重用根据状态转换图生成的，满足（基本）状态/转换覆盖的测试用例（包括关于起始状态和结束状态的规则）。

2. 构建状态转换表，并确认对于那些已定义的行都有对应的测试用例。不然，就是现有的测试用例有问题，或者状态转换表不正确，又或者状态转换图画错了。除非您已经找到问题所在并解决了问题，例如有必要的话可重新创建状态转换表或重新设计测试用例，不然不要继续以下的步骤。

3. 选择一个测试用例，这个测试用例覆盖的一个状态在转换表中有一行或多行未定义。修改测试用例，尝试触发未定义的事件/条件组合。注意，此时的操作域是未定义的。

4. 修改了测试用例之后，将对应的行标记为已覆盖。最方便的方法是将表格打印出来，再标记已覆盖的行。

5. 重复第 3 步、第 4 步，直到覆盖所有行。

再次说明，我们这里设计的是逻辑测试用例。

图 4-42 显示了如何根据已有的电子商务程序的测试用例，设计基于转换表的测试用例。在图的上方，您可以看到我从已有的测试用例中选择了一个测试用例：

（浏览、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。

- 已有的测试用例。
 - （浏览、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。
- 修改后的测试用例（只是为了覆盖那些未定义的“浏览”事件/条件）。
 - （浏览、尝试：继续购物、操作未定义、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。
 - （浏览、尝试：结账、操作未定义、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。
- 不要试图在同一个测试用例内的多个状态上覆盖各种未定义的事件/条件组合，因为你不知道到时候系统是否还是可测试的。
- 最理想的处理方式是系统能忽略未定义的事件/条件组合，或者能提供一个有用的错误消息，同时系统能继续正常地运行下去。

图 4-42 设计测试用例示例

现在，我开始修改测试用例以覆盖那些未定义的“浏览”事件/条件。

一个测试用例是（浏览、尝试：继续购物、操作未定义、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。

另一个测试用例是（浏览、尝试：结账、操作未定义、加入购物车、选择对话框、结账、登录对话框、登录[正确]、交易对话框、放弃、<无操作>、退出）。

对于“浏览”这个状态，还有其他 6 个修改后的测试用例，不过我并未在图中列出。正如您所看到的，设计这些测试用例是一个机械化的过程。只要您足够小心，并记住每一个您已经覆盖的行，比如使用我曾提到过的标记，那么您就不太可能会漏掉某一个测试用例。

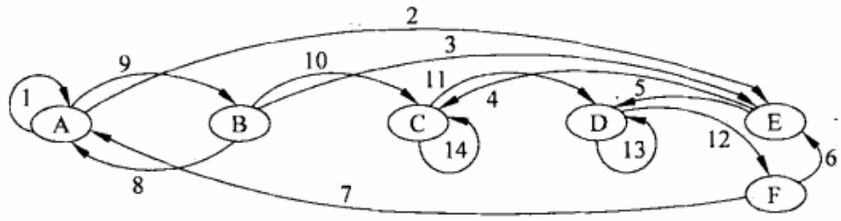
您可能注意到我在每一个测试步骤中只覆盖到了一个未定义的事件/条件组合。为什么？因为等价类划分技术存在一种规则，即我们不应该将多个无效值组合在同一个无效测试用例中。在这个例子中，每一行仅对应一个无效值。如果我们在一个测试步骤内覆盖两行，那么我们无法确保系统在处理过第一个无效值之后依旧可测。

注意，我说过操作是未定义的。那么，在这些条件下系统的理想行为是什么呢？最理想的处理方式是系统能忽略未定义的事件/条件组合，或者如果能提供一个有用的错误消息那就更好了。同时，系统能够继续正常地运行下去。在未从业务分析师、需求规格说明、系统设计师或其他权威那里得到任何有意义的输入的情况下，我会将任何其他输出作为一个缺陷，包括某些谜一样的错误消息，例如“刚刚发生了不可能发生的事情。”（这并不是我编造出来的。有一次一个 RBCS 课程的学员告诉我，当她在输入一个未定义的值时，就遇到了这样的错误消息）。

pin51.com 最新最全源分享下载 拼吾爱

4.2.24 切换覆盖

图 4-43 展示了我们应该如何使用切换覆盖的概念去设计转换序列。我将用电子商务的例子来讲解这个概念。



0-切换			1-切换								
A1	A2	A9	A1A1	A1A2	A1A9				A9B10	A9B8	A9B3
B10	B8	B3	B10C14	B10C11	B10C4	B8A1	B8A2	B8A9			
C14	C11	C4	C14C14	C14C11	C14C4	C11D13	C11D12	C11D5			
D13	D12	D5	D13D13	D13D12	D13D5	D12F6	D12F7				
F6	F7					F7A1	F7A2	F7A9			

图 4-43 N-1 切换覆盖示例

在图 4-43 的上方，您看到的这个状态转换图和先前的图是一样的，只不过我用字母替代了状态标签，用数字替代了转换标签。现在，一个状态/转换对可以写成一个字母加上一个数字。注意，我之所以没有在图下部的表格中列出数字后面的字母，是因为对于给定一个转换，其后的状态是明确的。对于一个给定的数字，只有唯一一个箭头与其对应，这个箭头指向一个标有特定字母的状态，且一个箭头只能指向一个状态。

这个表格有两列。第一列是我们必须覆盖到的状态/转换对，以满足 0-切换覆盖。花点时间仔细看看，如果您设计的测试用例覆盖了 0-切换列中的每一个状态/转换对，您的测试也就满足了我们曾提到过的（基本）状态/转换覆盖。

确定 0-切换列的过程是很简单的。在第一行中，每一个从初始状态出发的转换都对应一格。从图中可以看到，从 A 状态出发的转换有 3 个。然后对每一个有向外转换的状态，重复这个过程。不过注意，E 状态没有对应的行，因为 E 是结束状态，没有向外的转换。还有，在这个例子中，几乎每个状态都有 3 个向外的转换。

构建 1-切换列中的内容就有点困难了，不过如果您非常仔细地观察，那么依旧可以发现一些规律。注意，其实在每次 0-切换中的转换完成之后，我们是处在一个新的状态中，只不过我们并未在 0-切换列的单元格中明确标示出这个状态而已。我们曾经提到过，对于每个状态最多有 3 个向外的转换。这意味着，对于本例而言，每个 0-切换单元格最多可以拓展到 3 个 1-切换单元格。

因此，我们可以将 A 行中的每个 0-切换列单元格复制到 A 行中 1-切换列的 3 个单元格中，共产生 9 个新单元格。然后仔细思考，对于 A 行 1-切换列中每 3 个一组的单元格，下一步我们将有可能处于哪一个状态。最后再参考可能对应的 0-切换列的单元格，我们就能确定 1-切换列中单元格的剩余内容。

注意，1-切换列中的空单元格表示我们第一次转换之后进入的那个状态没有向外的转换。在本例中，图 4-43 中的 E 状态，即完整图中的“退出”状态，没有外向转换。

那么，在给定一组状态/转换序列的情况下，无论是 0-切换、1-切换、2-切换甚至更高级的切换，我们应该怎样设计测试用例去覆盖这些序列，并达到我们期望的覆盖级别？我还是将重用已有的用于满足（基本）状态/转换覆盖或 0-切换覆盖的测试用例。

1. 重用根据状态转换图生成的，满足（基本）状态/转换覆盖的测试用例（包括关于起始状态和结束状态的规则）。

2. 使用我们刚才介绍的技术构建切换表，并确认已有的测试用例覆盖了 0-切换列中的每个单元格。不然，就是您现有的测试用例有问题，或者您的切换表不正确，又或者您的状态转换图画错了。除非您已经找到问题所在并解决了问题，例如，有必要的话可重新创建切换表或重新设计测试用例，不然不要继续以下的步骤。完成上述检查后，再检查测试用例覆盖的更高级别的切换。

3. 现在，设计一个测试用例，这个测试用例覆盖的切换序列的最后一个状态可能是一个更高级别的切换序列的起始状态。在这个过程中根据需要可以使用已有的 0-切换序列。接下来，在测试用例中覆盖这个更高级别的切换序列，并再次检查测试用例的最终状态是哪一个。在理想情况中，这个新的最终状态将是另一个更高级别的转换序列的起始状态，但如果不是的话，可以使用 0-切换序列。您可以根据状态转换图，仔细地设计测试用例去覆盖那些更高级别的序列。重复这个过程直到测试用例必须结束。

4. 在设计测试用例时，一旦您在测试用例中覆盖了切换序列，那么就将这个切换序列标记为已覆盖。简单的做法是打印一份切换表，然后覆盖一个切换，就标记其对应的单元格。

5. 重复第 3 步和第 4 步，直到覆盖到了所有的切换序列。

再次说明，在这个过程中设计的是逻辑测试用例。

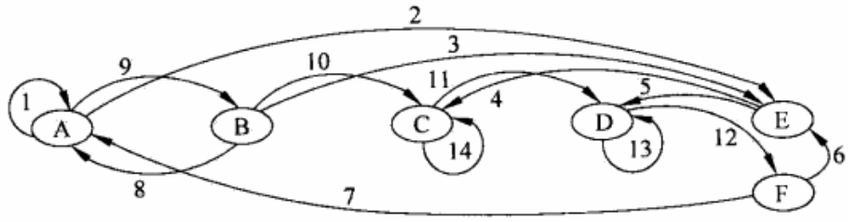
图 4-43 阐述了对（基于状态测试用例设计的）推导技术在之前提到的电子商务案例中的应用。在完成了上述第 2 步之后，即通过评估在满足 0-切换覆盖过程中所满足的覆盖，我们可以看到切换表的大部分都已变成了阴影。那些蓝色阴影的单元格覆盖了现有的 5 个（基本）状态/转换覆盖测试。

现在，我们再增加 5 个测试用例以满足 1-切换覆盖，如图 4-44 中所示。表格中红色的单元格表示由这 5 个新的 1-切换覆盖测试用例所覆盖到。

- （A1A1A2）。
- （A9B8A1A9B8A2）。
- （A9B10C14C14C4）。
- （A9B10C11D13D13D5）。
- （A9B10C11D12F7A1A9B10C11D12F7A9）。

设计高级别切换覆盖测试用例的算法和先前提到的设计行覆盖测试用例的算法都重用了满足（基本）状态/转换覆盖的测试用例。从测试设计的角度看这是高效的。但从测试执行的角度看，这种做法同时也是保守的，因为我们先覆盖了简单的东西，然后再覆盖难的东西。

pin51.com 最新最全源代码 拼吾爱



0-切换			1-切换								
A1	A2	A9	A1A1	A1A2	A1A9				A9B10	A9B8	A9B3
B10	B8	B3	B10C14	B10C11	B10C4	B8A1	B8A2	B8A9			
C14	C11	C4	C14C14	C14C11	C14C4	C11D13	C11D12	C11D5			
D13	D12	D5	D13D13	D13D12	D13D5	D12F6	D12F7				
F6	F7					F7A1	F7A2	F7A9			

图 4-44　设计测试示例

然而，无论在行覆盖或是 1-切换覆盖测试中，如果我们从零开始设计测试用例，那么很有可能可以减少测试用例的数目。如果在测试过程中最重要的事情是测试用例要尽可能少，那么您应该去寻找那些能减少已创建的测试用例的方法，或者修改我们刚才讲到的设计规程，从头开始设计，而不是重用已有的 0-切换的测试用例。

4.2.25　用其他技术进行状态测试

在结束基于状态的测试之前，我们再看看几个有趣的问题。

首先，将等价类划分技术和边界值分析技术和基于状态的测试技术组合在一起的效果如何？答案是，很好。

在电子商务这个例子中，假设最小的交易金额为 10 美元，最大为 10 000 美元。在这种情况下，如图 4-45 所示，我们可以对交易[正确]和交易[错误]事件/条件组合进行边界值分析。在设计测试时不仅要覆盖转换、行、或转换序列，而且也要覆盖边界值，这样我们就不得不尝试不同的交易数目。

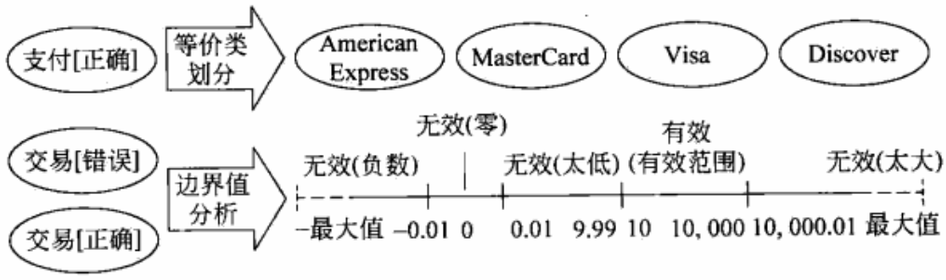


图 4-45　等价类划分和边界值分析

我们也可以对支付[正确]事件/条件组合进行等价类划分。例如，假设我们能接受 4 种不同类型的信用卡。在设计测试时不仅要覆盖转换、行、或转换序列，而且也要覆盖等价类划分，这样我们就不得不尝试不同的支付类型。

现在回到我开始讨论这 3 种业务逻辑测试技术的时候，我曾问过的一个问题，什么

时候我们应该使用决策表，什么时候我们应该使用状态图？

在某些情况下，这只是取决于个人的喜好。决策表方便简洁。如果我们并不关心高级别的覆盖或测试中状态的效果，那么在很多受状态影响的场景中，我们也可以使用决策表技术，用条件来模仿状态。但是，如果决策表的条件部分变得太大了，那么可能就不太适合了。记住，使用基于状态的技术测试覆盖一般会更高。在大多数情况下您应该能区分哪个技术更好。不过，如果觉得还是不明白到底该用哪个，那么可以两种技术都试试，然后选择更适合您的那个。

4.2.26 状态测试练习

该练习包含 4 个部分：

1. 使用在上一个练习中您设计的半正式用例，从电话银行经理的角度将这个用例转换成状态转换图。

2. 设计测试用例覆盖图中的状态和转换（0-切换）转换。

3. 根据状态转换图画状态转换表，并根据状态转换表设计其它错误处理测试用例。

4. 设计 1-切换级别的切换表，并增加额外的测试用例以满足 1-切换覆盖。

这 4 部分的答案在下面。在您查看下一部分的答案前，应该先查看前一部分的答案，以和自己的答案对照（并且，有必要的话，修改自己的下一部分的答案）。

4.2.27 状态测试练习解答

根据表 4-3 中的半正式用例生成的状态转换图如图 4-46 所示。

现在我们规定任何测试用例必须在初始的等待状态开始，并且只能在等待状态或换班状态结束。下列测试用例能满足状态和转换覆盖：

1.（等待、电话、贷款界面、超额[价值]、移交、授权、继续、系统列出贷款、贷款列表、提供保险、保险界面、顾客接受保险、加入到贷款套餐、顾客接受贷款，发送给 LoDoPS、等待）

2.（等待、电话、贷款界面、超额[贷款]、移交、授权、继续、系统列出贷款、贷款列表、提供保险、保险界面、顾客拒绝保险、归档、顾客接受贷款，发送给 LoDoPS、等待）

3.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、系统拒绝保险、归档、顾客拒绝贷款、归档、等待）

4.（等待、电话、贷款界面、超额[贷款]、移交、系统拒绝贷款、归档、等待）

5.（等待、电话、贷款界面、系统拒绝贷款、归档、等待）

6.（等待、电话、贷款界面、顾客取消、归档、等待）

7.（等待、电话、贷款界面、超额[贷款]、移交、顾客取消、归档、等待）

8.（等待、电话、贷款界面、系统列出贷款、贷款列表、顾客取消、归档、等待）

9.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、顾客取消、归档、等待）

pin51.com 拼吾爱 最新最全源分享下载站

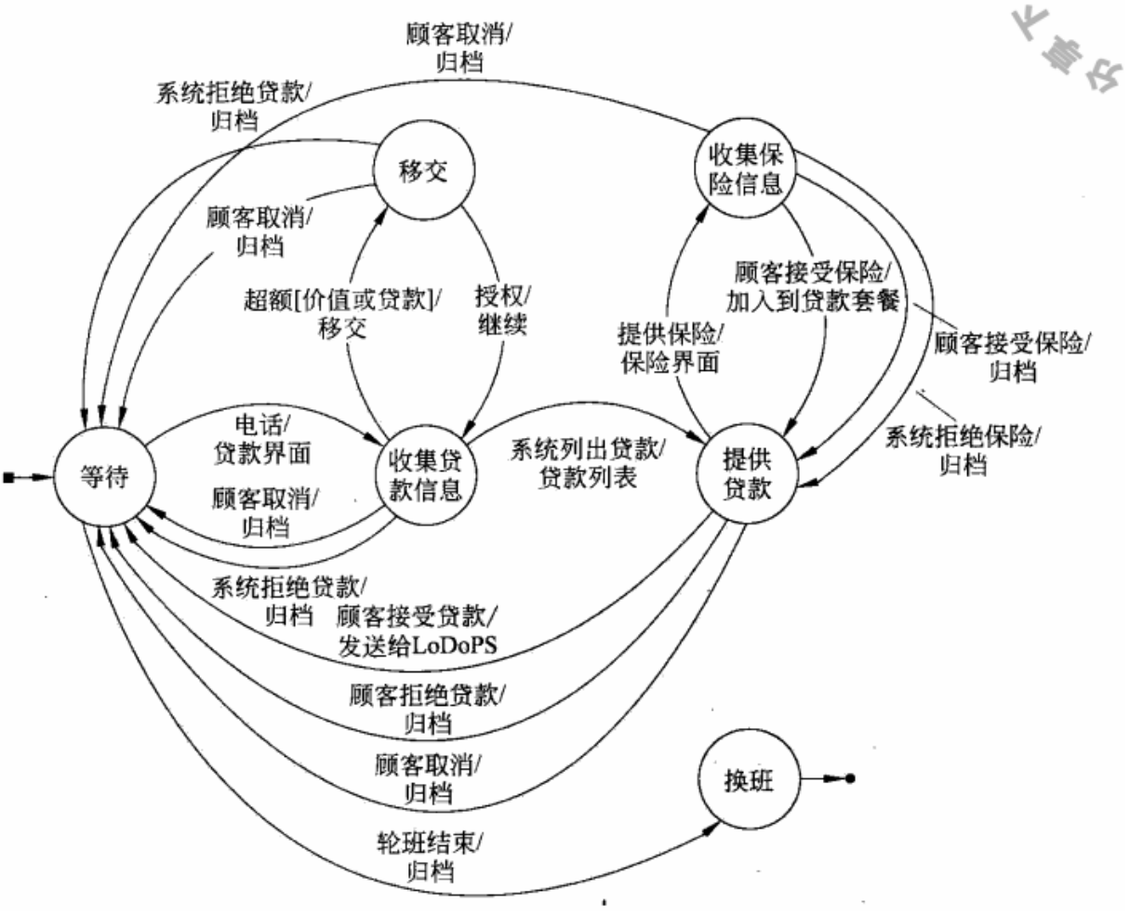


图 4-46 HELLOCARMS 状态转换图

10.（等待、轮班结束、登出、换班）

注意，这里我并没有对贷款额度或房产价值进行等价类划分和边界值分析，虽然我本可以这么做的。

接下来要生成状态转换表。这里一共有 6 个不同状态：

1. 等待。
2. 收集贷款信息。
3. 移交。
4. 提供贷款。
5. 收集保险信息。
6. 换班。

有 13 个不同的事件/条件组合：

1. 电话。
2. 超额[价值或贷款]。
3. 授权。
4. 系统拒绝贷款。
5. 系统列出贷款。
6. 提供保险。

pin51.com 新客发 新客免费资源分享下载

7. 顾客接受保险。
8. 顾客拒绝保险。
9. 系统拒绝保险。
10. 顾客接受贷款。
11. 顾客拒绝贷款。
12. 顾客取消。
13. 轮班结束。

因此，在这个状态转换表中有 78 行，如表 4-10 所示。

表 4-10 HELLOCARMS 状态转换表

#	当前状态	事件/条件	操作	新状态
	等待	电话	贷款界面	收集贷款信息
		超额[价值或贷款]	未定义	未定义
		授权	未定义	未定义
		系统拒绝贷款	未定义	未定义
		系统列出贷款	未定义	未定义
		提供保险	未定义	未定义
		顾客接受保险	未定义	未定义
		顾客拒绝保险	未定义	未定义
		系统拒绝保险	未定义	未定义
		顾客接受贷款	未定义	未定义
		顾客拒绝贷款	未定义	未定义
		顾客取消	未定义	未定义
		轮班结束	未定义	未定义
	收集贷款信息	电话	未定义	未定义
		超额[价值或贷款]	移交	移交
		授权	未定义	未定义
		系统拒绝贷款	归档	等待
		系统列出贷款	贷款列表	提供贷款
		提供保险	未定义	未定义
		顾客接受保险	未定义	未定义
		顾客拒绝保险	未定义	未定义
		系统拒绝保险	未定义	未定义
		顾客接受贷款	未定义	未定义
		顾客拒绝贷款	未定义	未定义
		顾客取消	归档	等待
		轮班结束	未定义	未定义

pin51.com 拼吾爱 最新最全软件下载

续表

#	当前状态	事件/条件	操作	新状态
	移交	电话	未定义	未定义
		超额[价值或贷款]	未定义	未定义
		授权	继续	收集贷款信息
		系统拒绝贷款	归档	等待
		系统列出贷款	未定义	未定义
		提供保险	未定义	未定义
		顾客接受保险	未定义	未定义
		顾客拒绝保险	未定义	未定义
		系统拒绝保险	未定义	未定义
		顾客接受贷款	未定义	未定义
		顾客拒绝贷款	未定义	未定义
		顾客取消	归档	等待
		轮班结束	未定义	未定义
	提供贷款	电话	未定义	未定义
		超额[价值或贷款]	未定义	未定义
		授权	未定义	未定义
		系统拒绝贷款	未定义	未定义
		系统列出贷款	未定义	未定义
		提供保险	保险界面	收集保险信息
		顾客接受保险	未定义	未定义
		顾客拒绝保险	未定义	未定义
		系统拒绝保险	未定义	未定义
		顾客接受贷款	发送给 LoDoPS	等待
		顾客拒绝贷款	归档	等待
		顾客取消	归档	等待
		轮班结束	未定义	未定义
	收集保险信息	电话	未定义	未定义
		超额[价值或贷款]	未定义	未定义
		授权	未定义	未定义
		系统拒绝贷款	未定义	未定义
		系统列出贷款	未定义	未定义
		提供保险	未定义	未定义
		顾客接受保险	加入到贷款套餐	提供贷款
		顾客拒绝保险	归档	提供贷款
		系统拒绝保险	归档	提供贷款
		顾客接受贷款	未定义	未定义

续表

#	当前状态	事件/条件	操作	新状态
	收集保险信息	顾客拒绝贷款	未定义	未定义
		顾客取消	归档	等待
		轮班结束	未定义	未定义
	换班	电话	未定义	未定义
		超额[价值或贷款]	未定义	未定义
		授权	未定义	未定义
		系统拒绝贷款	未定义	未定义
		系统列出贷款	未定义	未定义
		提供保险	未定义	未定义
		顾客接受保险	未定义	未定义
		顾客拒绝保险	未定义	未定义
		系统拒绝保险	未定义	未定义
		顾客接受贷款	未定义	未定义
		顾客拒绝贷款	未定义	未定义
		顾客取消	未定义	未定义
		轮班结束	未定义	未定义

为了达到行覆盖，增加了下列测试用例：

11.（等待、尝试：超额[价值或贷款]、操作未定义、轮班结束、登出、换班）

12.（等待、尝试：授权、操作未定义、轮班结束、登出、换班）

13.（等待、尝试：系统拒绝贷款、操作未定义、轮班结束、登出、换班）

14.（等待、尝试：系统列出贷款、操作未定义、轮班结束、登出、换班）

15.（等待、尝试：提供保险、操作未定义、轮班结束、登出、换班）

16.（等待、尝试：顾客接受保险、操作未定义、轮班结束、登出、换班）

17.（等待、尝试：顾客拒绝保险、操作未定义、轮班结束、登出、换班）

18.（等待、尝试：系统拒绝保险、操作未定义、轮班结束、登出、换班）

19.（等待、尝试：顾客接受贷款、操作未定义、轮班结束、登出、换班）

20.（等待、尝试：顾客拒绝贷款、操作未定义、轮班结束、登出、换班）

21.（等待、尝试：顾客取消、操作未定义、轮班结束、登出、换班）

22.（等待、尝试：轮班结束、操作未定义、轮班结束、登出、换班）

23.（等待、电话、贷款界面、尝试：电话、操作未定义、顾客取消、归档、轮班结束、登出、换班）

24.（等待、电话、贷款界面、尝试：超额[价值或贷款]、操作未定义、顾客取消、归档、轮班结束、登出、换班）

25.（等待、电话、贷款界面、尝试：授权、操作未定义、顾客取消、归档、轮班结束、登出、换班）

pin51.com 最新最全源码分享 拼吾爱

26.（等待、电话、贷款界面、尝试：提供保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

27.（等待、电话、贷款界面、尝试：顾客接受保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

28.（等待、电话、贷款界面、尝试：顾客拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

29.（等待、电话、贷款界面、尝试：系统拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

30.（等待、电话、贷款界面、尝试：顾客接受贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

31.（等待、电话、贷款界面、尝试：顾客拒绝贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

32.（等待、电话、贷款界面、尝试：轮班结束、操作未定义、顾客取消、归档、轮班结束、登出、换班）

33.（等待、电话、贷款界面、超额[贷款]、移交、尝试：电话、操作未定义、顾客取消、归档、轮班结束、登出、换班）

34.（等待、电话、贷款界面、超额[价值]、移交、尝试：超额[价值]、操作未定义、顾客取消、归档、轮班结束、登出、换班）

35.（等待、电话、贷款界面、超额[价值]、移交、尝试：系统列出贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

36.（等待、电话、贷款界面、超额[价值]、移交、尝试：提供保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

37.（等待、电话、贷款界面、超额[价值]、移交、尝试：顾客接受保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

38.（等待、电话、贷款界面、超额[价值]、移交、尝试：顾客拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

39.（等待、电话、贷款界面、超额[价值]、移交、尝试：系统拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

40.（等待、电话、贷款界面、超额[价值]、移交、尝试：顾客接受贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

41.（等待、电话、贷款界面、超额[价值]、移交、尝试：顾客拒绝贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

42.（等待、电话、贷款界面、超额[价值]、移交、尝试：轮班结束、操作未定义、顾客取消、归档、轮班结束、登出、换班）

43.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：电话、操作未定义、顾客取消、归档、轮班结束、登出、换班）

44.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：超额[贷款]、操作未定义、顾客取消、归档、轮班结束、登出、换班）

45.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：授权、操作未定义、顾客取消、归档、轮班结束、登出、换班）

46.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：系统拒绝贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

47.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：系统列出贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

48.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：顾客接受保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

49.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：顾客拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

50.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：系统拒绝保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

51.（等待、电话、贷款界面、系统列出贷款、贷款列表、尝试：轮班结束、操作未定义、顾客取消、归档、轮班结束、登出、换班）

52.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：电话、操作未定义、顾客取消、归档、轮班结束、登出、换班）

53.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：超额[贷款]、操作未定义、顾客取消、归档、轮班结束、登出、换班）

54.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：授权、操作未定义、顾客取消、归档、轮班结束、登出、换班）

55.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：系统拒绝贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

56.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：系统列出贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

57.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：提供保险、操作未定义、顾客取消、归档、轮班结束、登出、换班）

58.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：顾客接受贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

59.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：顾客拒绝贷款、操作未定义、顾客取消、归档、轮班结束、登出、换班）

60.（等待、电话、贷款界面、系统列出贷款、贷款列表、提供保险、保险界面、尝试：轮班结束、操作未定义、顾客取消、归档、轮班结束、登出、换班）

61.（等待、轮班结束、登出、尝试：电话、操作未定义、换班）

62.（等待、轮班结束、登出、尝试：超额[价值]、操作未定义、换班）

63.（等待、轮班结束、登出、尝试：授权、操作未定义、换班）

64.（等待、轮班结束、登出、尝试：系统拒绝贷款、操作未定义、换班）

65.（等待、轮班结束、登出、尝试：系统列出贷款、操作未定义、换班）

66.（等待、轮班结束、登出、尝试：提供保险、操作未定义、换班）

pin51.com 最新最全源码分享 拼吾爱

67.（等待、轮班结束、登出、尝试：顾客接受保险、操作未定义、换班）
68.（等待、轮班结束、登出、尝试：顾客拒绝保险、操作未定义、换班）
69.（等待、轮班结束、登出、尝试：系统拒绝保险、操作未定义、换班）
70.（等待、轮班结束、登出、尝试：顾客接受贷款、操作未定义、换班）
71.（等待、轮班结束、登出、尝试：顾客拒绝贷款、操作未定义、换班）
72.（等待、轮班结束、登出、尝试：顾客取消、操作未定义、换班）
73.（等待、轮班结束、登出、尝试：轮班结束、操作未定义、换班）

注意，在每个测试用例的最后增加“轮班结束、登出、换班”这个序列是为了减少在测试过程中造成的错误条件影响后续结果的可能性。

正如先前所讲的那样，我们的这种设计方法是保守的，因为在每个测试用例中只尝试了一种未定义事件。在测试执行过程中，如果测试人员发现系统的健壮性很好并能忽略这些未定义事件，那么在接下来的测试中可以将这些测试用例与其他测试用例合并，并减少测试用例的数目。

现在，为了生成满足 1-切换覆盖的切换表，我首先用先前提到过的字母数字命名方法简化了现有的状态转换图。简化后的版本如图 4-47 所示。

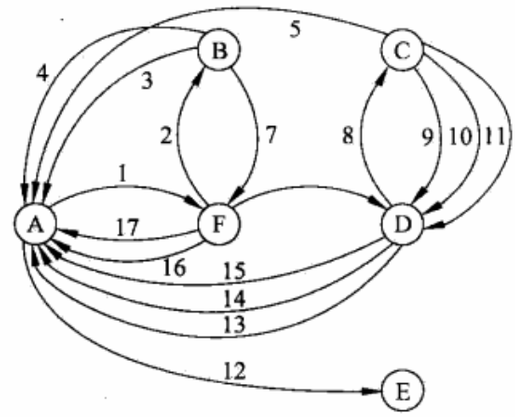


图 4-47 HELLOCARMS 系统 1-切换表格的状态转换图

根据该图生成的 1-切换表如表 4-11 所示。注意，在生成这个表格的时候我使用了从图中总结出的一些模式。例如，图中每个状态的向外转换的最大数目是 4，因此在 0-切换列和 1-切换列中我都安排了 4 个列。在一开始的时候，由于在图中共有 6 个状态，所以对于 0-切换列中的每一行，我在 1-切换列中安排了 6 行与其对应。建立这样一个稀疏的表格，是为了简化设计的过程。而且随着设计过程的进行，我们可以删除其中大部分空行。

表 4-11 HELLOCARMS 切换表

0-切换				1-切换			
A1	A12			A1F2	A1F6	A1F16	A1F17
B3	B4	B7		B3A1	B3A12		
				B4A1	B4A12		
				B7F2	B7F6	B7F16	B7F17
C5	C9	C10	C11	C5A1	C5A12		
				C9D8	C9D13	C9D14	C9D15
				C10D8	C10D13	C10D14	C10D15
				C11D8	C11D13	C11D14	C11D15
D8	D13	D14	D15	D8C5	D8C9	D8C10	D8C11
				D13A1	D13A12		

续表

0-切换				1-切换			
D8	D13	D14	D15	D14A1	D14A12		
				D15A1	D15A12		
F2	F6	F16	F17	F2B3	F2B4	F2B7	
				F6D8	F6D13	F6D14	F6D15
				F16A1	F16A12		
				F17A1	F17A12		

现在，有了这个切换表，我们就能对照着看看已有的测试用例覆盖了其中的哪些部分。注意，为了满足行覆盖而设计的测试用例不能计算在内，因为您不能假设在一个无效事件之后的情况是典型的。

下列是我们已有的测试用例，为了更方便与切换表核对，所以我们用字母数字命名方法重写了一遍。

1. (A1F2B7F6D8C9D15)。
2. (A1F2B7F6D8C10D15)。
3. (A1F6D8C11D14)。
4. (A1F2B4)。
5. (A1F16)。
6. (A1F17)。
7. (A1F2B3)。
8. (A1F6D13)。
9. (A1F6D8C5)。
10. (A12)。

覆盖分析结果如表 4-12 所示。

表 4-12 满足 0-切换覆盖的 HELLOCARMS 切换表

0-切换				1-切换			
A1	A12			A1F2	A1F6	A1F16	A1F17
B3	B4	B7		B3A1	B3A12		
				B4A1	B4A12		
				B7F2	B7F6	B7F16	B7F17
C5	C9	C10	C11	C5A1	C5A12		
				C9D8	C9D13	C9D14	C9D15
				C10D8	C10D13	C10D14	C10D15
				C11D8	C11D13	C11D14	C11D15
D8	D13	D14	D15	D8C5	D8C9	D8C10	D8C11
				D13A1	D13A12		
				D14A1	D14A12		
				D15A1	D15A12		

pin51.com 最新最全源码下载 拼吾爱

续表

0-切换				1-切换			
F2	F6	F16	F17	F2B3	F2B4	F2B7	
				F6D8	F6D13	F6D14	F6D15
				F16A1	F16A12		
				F17A1	F17A12		

为了满足1-切换覆盖，我新增了一些测试用例，如表4-13所示。为了方便起见，这里我使用的是字母数字命名方法，当然，最后我还是要就将这些测试用例转换成逻辑测试用例的形式，有起始状态和结束状态，中间是一系列的事件和操作。

表4-13 满足0-切换和1-切换覆盖的HELLOCARMS切换表

0-切换				1-切换			
A1	A12			A1F2	A1F6	A1F16	A1F17
B3	B4	B7		B3A1	B3A12		
				B4A1	B4A12		
				B7F2	B7F6	B7F16	B7F17
C5	C9	C10	C11	C5A1	C5A12		
				C9D8	C9D13	C9D14	C9D15
C5	C9	C10	C11	C5A1	C5A12		
				C10D8	C10D13	C10D14	C10D15
				C11D8	C11D13	C11D14	C11D15
D8	D13	D14	D15	D8C5	D8C9	D8C10	D8C11
				D13A1	D13A12		
				D14A1	D14A12		
				D15A1	D15A12		
F2	F6	F16	F17	F2B3	F2B4	F2B7	
				F6D8	F6D13	F6D14	F6D15
				F16A1	F16A12		
				F17A1	F17A12		

74.（A1F2B3A1F16A1F2B3A12）

75.（A1F2B4A1F2B4A12）

76.（A1F2B7F2B7F16A12）

77.（A1F2B7F2B7F17A12）

78.（A1F17A1F6D14）

79.（A1F6D14）

80.（A1F6D8C5A1F6D8C5A12）

81.（A1F6D8C9D8C9D13A12）

82.（A1F6D8C10D8C9D14A1F6D15A1F6D14A12）

pin51.com 最新最全资源分享下载

83.（A1F6D8C10D8C10D14）

84.（A1F6D8C10D13A1F17）

85.（A1F6D8C10D15A12）

86.（A1F6D8C11D8C11D14)

注意，在大部分测试中，测试用例结束时会回到等待状态，然后在未登出的情况下，开始一个新的测试用例。如果很在意测试执行的效率，可以将上述的某些测试用例与一开始的测试用例压缩在一起，只要在不登出的情况下连续执行前 10 个测试用例即可。

注意在 1-切换测试用例中有两个工作流可能是不现实的：

- 不止一次向高级电话银行经理移交申请以获得授权。
- 在同一个申请过程中，不止一次向顾客提供保险。

您可能需要询问业务分析师或系统设计师，系统是否可以这么做。

4.2.28 结对测试

对于我们讨论过的上述 3 种技术，决策表、用例和基于状态的测试，其中各种因素的交互都是可辨别、可分析的。对于决策表和因果图，我们可以看到条件以及系统操作之间的交互。对于用例，我们发现系统可以正常处理某些条件，而某些条件则会导致异常。对于基于状态的测试，根据其所在状态，事件和条件会触发系统操作或状态的转换。

那么当我们很难确定各种因素之间的交互，甚至根本就没有交互的时候应该怎么办呢？例如，您要测试一个基于浏览器的程序与环境之间的兼容性，您有 4 个主要的环境因素，不同类型和版本的浏览器、不同的网络连接速度、不同的操作系统类型、版本和补丁等级、不同的病毒和木马软件扫描程序，每一个因素都有近一打可选项。无论面对什么样的配置，您的程序都应该能正确运行，但确实是这样么？唯一能确定的方法是测试所有的配置，不过这是不可能的。因为如果您有 4 个因素，每个因素有一打选项，那么您将面对 12 乘以 12 乘以 12 乘以 12 共 20 736 种不同配置。

当因素的各个选项可不受限制组合的时候，结对测试技术可用于控制组合爆炸。概念上而言，在结对测试中，我们确保每个选项至少在一个配置中覆盖到，每一对选项至少在一个配置中覆盖到，以及在所有配置中每一个选项和每一对选项覆盖到的次数相近。在结对测试中，不会覆盖到所有高阶的组合，例如三元组、四元组、五元组等。

我刚才所说的“不受限制的选项”是指，一般而言，选项之间是相互独立的。任何因素的任何选项可以和任何其他因素的任何选项同时存在。一个典型的例子是配置测试。在同一台电脑上，不同版本的软件可以共存并正确交互。任何不兼容的情况就是一个缺陷。

那么，结对测试的基本模型又是什么呢？有两个基本模型。一个叫做正交阵列，还有一个叫做所有选项对表。这两个模型都是表格。在表格中的每一行代表了在测试配置中要包含的特定选项。我们可以直接创建表格（例如所有选项对表），或者通过将要解决

pin51.com 最新最全 拼吾爱

的测试问题映射到已有的表格中从而创建一个新表格（例如正交阵列）。根据这两个表格的特性，它们能保证每一个因素的所有选项至少覆盖一次，并且覆盖任意两个因素中的所有选项对。

对于这些表格的覆盖准则是很明确的，即：每一行至少应该在一个测试用例中覆盖到。满足这个覆盖准则，就意味着我们对每一个因素的所有选项至少测试了一次，并且测试了任意两个因素中的所有选项对。

这里的缺陷假设是：这样的覆盖级别（对您的项目）已经足够。换句话来说，如果存在问题，那么我们可以通过单个选项或一对选项触发其中大部分的问题。而对于其他特定于3个选项、4个选项或者更高级组合的问题，发生的可能性很小。

正如我刚才提到过，结对测试有两种基本方法。其中第一种是使用正交阵列。在因特网或者各种书中有很多这方面的例子。例如，在 www.research.att.com/~njas/oadir 这个链接中有一个正交阵列库。由于这些阵列是 AT&T 公司用于测试电话系统的，所以有一定的可靠性。或者，您可以用一个工具创建一个所有选项对表。这样的工具在因特网上有不少。在 www.pairwise.org 这个网站上您可以找到一个用于结对测试的工具的列表，免费的或商业的工具都有。

正交阵列表和所有选项对表之间有一点细微的差别。这两种表的行数通常是不同的，同时每一个选项对出现的次数也是不同的。不过对于兼容性测试而言，这种差别通常并不重要。

在本书中，我将介绍如何使用正交阵列表。如果您也需要用到这种表格，可以跟我一起做。

图 4-48 给出的是一个最简单的正交阵列表。共有两个因素，因素 1 和因素 2。每个因素有两个选项，0 和 1。正如您所看到的，该正交阵列表覆盖到了每一个因素的每一个选项，而且，两个因素的任意一对选项对都对应着表中的一行（只有一行）。

	因素	
测试用例	1	2
1	0	0
2	0	1
3	1	0
4	1	1

图 4-48　一个简单的正交阵列

好吧，这没什么特别的。我们有两个选项，4 行，2 乘以 2 等于 4。到目前为止，正交阵列没什么吸引人的地方。

图 4-49 就比较有意思了。这个正交阵列比先前那个稍大一点。有 3 个因素。每个因素两个选项。不过最终的结果还是只有 4 行。

发生了什么？我们是不是漏掉了一些选项对？不，并不是这样。您可以注意到，如先前一样，每一对因素还是有 4 对选项对。现在，我们一一列举这些对：

- 对于第一个因素对，因素 1 和因素 2，我们有 00、01、10、11 分别对应于行 1、2、3、4。

pin51.com 拼客发 资源分享

	因素		
测试用例	1	2	3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

图 4-49 大一点的正交阵列

- 对于第二个因素对，因素 1 和因素 3，我们有 00、01、11 和 10，分别对应于行 1、2、3、4。这 4 对选项与第一次的 4 对选项对应着不同的行。
- 对于第三个因素对，因素 2 和因素 3，我们有 00、11、01 和 10，分别对应于行 1、2、3、4。同样，这次的 4 对选项与上一次的 4 对选项所对应的行又有所不同。

如果我们希望测试所有三元组，那么这个表格应该有 8 行。但是，在结对测试中，我们明确放弃了这种级别的覆盖。我们假设只要覆盖到每个因素中的每个选项以及每一对因素中的所有可能的选项对就已经足够了。

我们刚讨论的这个正交阵列还是没什么特别的。我们只能处理 3 个因素的情况，在某些情况下这可能已经足够了，但是更重要的是，我们只能处理每个因素两个选项的情况。那么我们应该如何根据特定的测试问题来选择合适的正交阵列呢？

有 3 条规则可以指导您如何选择一个正交阵列：

- 首先，正交阵列中的列数不能少于因素的个数。不过，实际上您经常会发现列数太多了。这没什么问题。如果列太多，您只要删掉多余的列即可。
- 第二，每一列中应该有足够的空间存放每个因素的各个选项。在前面的那个例子中只有两个数字，0 和 1。如果您有一个因素有 3 个选项，那么就应该至少有 3 个数字，0、1 和 2。再次说明，您常常会发现一个阵列中有过多的数字，这也不是什么问题，（通常）那些多余的数字并不对应任何选项，因此您可以用该因素的任何有效选项替代这个多余的数字。我们常称这种情况为“测试人员自选”，并标识为波浪号～。
- 第三，行数不小于两个最大的选项数的乘积。例如，一个因素有 4 个选项、一个因素有 3 个选项还有一个因素只有 2 个选项，那么，您的阵列表至少应该有 4 乘以 3，即 12 行。如果您经常发现阵列中的行太多，那么我们就有麻烦了。如果行太多，您不能直接将包含着选项对的行删除而只能仔细地检查这些行，每次检查两行，看看能否通过使用“测试人员自选”这个选项将这两行合并。

您可以通过一个包含 6 个步骤的过程将您的测试问题映射到已有的正交阵列表中。映射的过程是完全机械化的，因此只要用 Excel 或 Word 就能很方便地做到。您只要将下载的包含正交阵列的文本文件导入到 Excel 或 Word 中，并按照下列 5 个步骤做就可以了：

1. 首先，删除所有多余的列。
2. 第二，在列的头部填入所有因素。
3. 第三，每次选择一列，然后将那个因素的选项映射到数字上。也就是说，用那个因素的第一个选项替代 0，用第二个选项替代 1，以此类推。您可以使用 Word 或 Excel

pin51.com 最新最全源码分享 拼吾爱

软件的查找替换功能来简化这个过程。映射完成之后如果列中还有其他数字，那么用波浪号替代这些数字，表示“测试人员自选”。

一般而言您做到这一步就可以了，不过此时阵列表中的行要比您所需要的多。如果您用于测试每个配置的时间大于一个小时，并且在测试执行的过程中您也没有其他的事做，那么可以继续第 4 步。

4. 在第 4 步中，删除那些不包含有用的单个选项或选项对的行。也就是说，删除那些都是波浪号的行。对于任意的两行，如果其中一行是波浪号而另一行是选项，或者两行中的选项都是相同的，那么您可以将这两行合并在一起。

5. 接下来，您可以马上进行第 5 步，或者您也可以等到测试执行的时候再这么做。对于表中任何多余的单元格，即那些含有波浪号的格，您可以指定其选项，以便更方便地测试、或者覆盖一些流行的配置、或者其他原因。正如我说的，这是测试人员自选的。

顺便提一下，如果您选择了一个错误的正交阵列，那么在映射的过程中，您很快就能发现问题。其中一个警示标志便是由于缺少足够的列或选项所以映射失败。另一个不是那么明显的标志是，在很多行中大多数单元格是波浪号，甚至全是波浪号。

对于这个过程我们已经进行了足够多的讨论，现在让我们来看看一个例子。让我们回到 www.rbcs-us.com 网站测试的例子。这次，假设我们要在功能测试的过程中进行兼容性测试，确保程序在一些环境配置中能正常运行。我们关注 4 个因素，每个因素有 2 个、3 个或者 4 个选项：

- 第一个因素是网络连接速度。我们有两个选项：拨号（DU）和宽带（BB）。
- 第二个因素是操作系统。我们有 4 个选项：Mac、Linux、Windows XP 和 Windows Vista。
- 第三个因素是安全设置。我们有 4 个选项：只使用本地操作系统的安全功能、Symantec 公司的产品、Trend Micro 公司的产品、McAfee 公司的产品。
- 第四个因素是浏览器。我们有 3 个选项：Firefox、Internet Explorer 和 Opera。

因此，我们应该选择一个有 4 列的阵列，每一个因素一列。其中一列使用两个数字，另一列使用 3 个数字，其他两列使用 4 个数字。但是，由于通常情况下阵列中每列的数字都是对称的，因此最终我们得到的阵列可能是每列 4 个数字。由于两个选项最多的因素，安全设置和操作系统，各有 4 个选项，因此阵列应该有 16 行。

在图 4-50 中，您看到的是我们选择的正交阵列以及映射后的结果。图左侧是从 AT&T 网站得到的阵列的文本内容。将这个文本文件存到硬盘上以后，再导入到 Excel，添加列标题，并且对每一列进行查找和替换操作，这个过程不会超过 5 分钟。到这里，映射过程就算完成了。由于这个阵列很完美，因此我们可以跳过第 4 步，不再合并各行。

花一点时间检查一下所有因素对的所有选项对。这里我们有 6 对因素：

1. 网络连接速度和操作系统。
2. 网络连接速度和安全设置。
3. 网络连接速度和浏览器。
4. 操作系统和安全设置。
5. 操作系统和浏览器。

pin51.com 软件开发资源分享下载

00000
01111
02222
03333
10123
11032
12301
13210
20231
21320
22013
23102
30312
31203
32130
33021

	因素			
测试用例	速度	操作系统	安全措施	浏览器
1	拨号	Mac	OS	Firefox
2	拨号	Linux	Symantec	IE
3	拨号	XP	Trend	Opera
4	拨号	Vista	McAfee	~
5	宽带	Mac	Symantec	Opera
6	宽带	Linux	OS	~
7	宽带	XP	McAfee	Firefox
8	宽带	Vista	Trend	IE
9	~	Mac	Trend	~
10	~	Linux	McAfee	Opera
11	~	XP	OS	IE
12	~	Vista	Symantec	Firefox
13	~	Mac	McAfee	IE
14	~	Linux	Trend	Firefox
15	~	XP	Symantec	~
16	~	Vista	OS	Opera

图 4-50　阵列选择和映射示例

6. 安全设置和浏览器。

在测试过程中，我们只要保证阵列中每一行的所对应的配置至少在一个测试用例中覆盖到即可。[5]

4.2.29　结对测试练习

假设您计划对 HELLOCARMS 系统最终的网络应用进行兼容性测试。Globobank 希望该系统能在下列配置中运行：

- 操作系统：Windows XP、Windows Vista 或者 Mac
- 浏览器：Internet Explorer（但只能在 Windows 操作系统上运行）、Firefox 或 Opera

同时，Globobank 希望 HELLOCARMS 系统能支持拨号和宽带这两种网络连接。

图 4-51、图 4-52 和图 4-53 是可能用到的正交阵列。

该练习包含 3 个部分：

1. 选择一个适当的阵列。
2. 将因素（操作系统、浏览器和网络连接速度）和每个因素的选项映射到阵列中。
3. 对那些有多余单元格的行以及配置无效的选项对进行处理。

5　对于结对测试技术的进一步讨论，尤其是正交阵列技术的起源和内在理论，请参考李-科普兰（Lee Copeland）的书 The Practitioner's Guide to Software Test Design 一书。

pin51.com 最新最全源分享下载 拼吾爱

	因素		
测试用例	1	2	3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

图 4-51 正交阵列 1

	因素			
测试用例	1	2	3	4
1	0	0	0	0
2	0	1	1	2
3	0	2	2	1
4	1	0	1	1
5	1	1	2	0
6	1	2	0	2
7	2	0	2	2
8	2	1	0	1
9	2	2	1	0

图 4-52 正交阵列 2

	因素				
测试用例	1	2	3	4	5
1	0	0	0	0	0
2	0	1	1	1	1
3	0	2	2	2	2
4	0	3	3	3	3
5	1	0	1	2	3
6	1	1	0	3	2
7	1	2	3	0	1
8	1	3	2	1	0
9	2	0	2	3	1
10	2	1	3	2	0
11	2	2	0	1	3
12	2	3	1	0	2
13	3	0	3	1	2
14	3	1	2	0	3
15	3	2	1	3	0
16	3	3	0	2	1

图 4-53 正交阵列 3

这三部分的答案在下面。在查看下一部分的答案前，应该先查看前一部分的答案，以和自己的答案对照（并且，有必要的话，修改自己下一部分的答案）。

4.2.30 结对测试练习解答

首先，我们要选择一个合适的正交阵列。因为我们有 3 个因素，每个因素最多有 3 个选项，所以我们应该选择一个有 3 列 9 行、每列 3 个数字的阵列；因此图 4-52 中的阵列比较合适。

接下来，我们将各因素和选项映射到阵列中，结果如表 4-14 所示。

pin51.com 新客发 最新最全源分享下载

表 4-14 映射后的 HELLOCARMS 正交阵列

	因 素		
测 试	操作系统	浏览器	速 度
1	XP	IE	拨号
2	XP	Firefox	宽带
3	XP	Opera	~
4	Vista	IE	宽带
5	Vista	Firefox	~
6	Vista	Opera	拨号
7	Mac	IE	~
8	Mac	Firefox	拨号
9	Mac	Opera	宽带

但在表 4-14 中有一个问题。需求中规定 Internet Explorer 只能在 Windows 系统上运行。而测试 7 将 Mac 和 Internet Explorer 组合在了一起。当然，由于阵列表的性质便是确保每个因素对的所有选项对至少出现一次，所以发生这种问题是必然的。

幸运的是，在该行中除了 Mac 操作系统与 Internet Explorer 浏览器这一无效的选项对之外，没有其他的选项对，只有一个波浪号表示“测试人员自选”，因此解决方法很简单，直接删除该行即可。最终的测试配置如表 4-15 所示。

表 4-15 HELLOCARMS 系统的最终测试配置

	因 素		
测 试	操作系统	浏览器	速 度
1	XP	IE	拨号
2	XP	Firefox	宽带
3	XP	Opera	~
4	Vista	IE	宽带
5	Vista	Firefox	~
6	Vista	Opera	拨号
7	Mac	Firefox	拨号
8	Mac	Opera	宽带

在结束对结对测试技术的讨论之前，有 4 个地方是我们需要注意的。

第一，注意我没有将表 4-15 称为正交阵列。这是因为在我删除了原来的第七行之后，这个表格不再满足正交阵列的性质。

第二，如果在查找替换过程中您选择的数字和选项之间的映射与我们不同，那么您可能会发现在无效的 Mac/Internet Explorer 行中，还有其他两个选项对。在这种情况中，您删除这一行之前，必须在其他行中覆盖这两个选项对，您可以使用测试人员自选符号或增加新的行。

pin51.com 最新最全 拼吾爱

第三，删除或合并行的时候要非常、非常小心。我们从因素对中选出特别的选项对。在您删除某一行之前，确保这一行中没有特别的选项对，例如未在其他行中覆盖的选项对。而在您合并两行之前，确保其中一行有些特定选项，例如非测试人员自选的选项，对应于另一行中的测试人员自选选项或者这两行中的这些特定选项相同。

最后，所有的选项对表可以变得更小，因此花点时间获取一个免费的软件工具帮助您进行结对测试还是值得的。在很多情况中，将选项映射到正交阵列的过程快速、方便，并且也很少有额外的行，因此您会倾向于更多地使用这种技术。

4.2.31 分类树

结对测试技术很方便，而且在上一部分中提到的兼容性测试在测试中也很常见。但是，有时候某些因素是受限制的，某个因素的一些选项不能与另一个因素的一些选项共存。而有时候，我们可能希望对某些因素或因素的组合进行更多的测试。例如，如果我们在测试一个程序的配置选项，那么其中的一些选项可能是相互影响的。

概念上而言，分类树能用于测试受限制的因素组合。同时这种技术也可以用于对某些因素进行重点测试。

分类树技术的基本模型是因素以及因素选项的图形表示，可以通过等价类划分技术生成这个模型。该技术还包含对因素和选项的组合规则的定义，包括某些因素应该满足的组合级别（例如，对于其中 3 个因素，要测试它们所有的三元组，而对于其他的因素，只要进行结对测试即可）。

根据这个模型，可以设计测试用例，列出各因素的选项组合。在实际使用的时候，需要一个工具帮助您进行设计。与正交阵列和所有选项对表一样，分类树技术的覆盖准则是表中的每一行应在至少一个测试用例中覆盖到。

> **ISTQB 术语**
>
> **分类树**：显示等价划分层次的树。该树在分类树方法中用于辅助测试用例的设计。
>
> **分类树方法**：运用分类树法而进行的一种黑盒测试设计技术，通过输入和/或输出域的组合来设计测试用例。

与结对测试一样，分类树技术的缺陷假设是该覆盖率级别（对您的项目）已经足够了。不同之处在于，在分类树技术中，根据因素的不同，覆盖率的级别是不同的。我们猜测某些因素的选项组合发生问题的可能性比较高，或者根据业务需求，某些因素的选项组合的风险级别比较高。我们可以将两个因素组合，或者 3 个因素组合，或者 4 个因素组合，甚至更高。当然，组合的级别越高，表越大，测试用例也就越多。

假设我们要测试 Microsoft Word 字体菜单，如图 4-54 所示。我们对下列因素和选项特别感兴趣，担心它们相互之间有不正确的影响。

- 类型（style）：regular、italic、bold、bold italic。
- 大小（size）：最小（8）、典型（12）、最大（72）。
- 删除线（strikethrough）：否，是，双线。

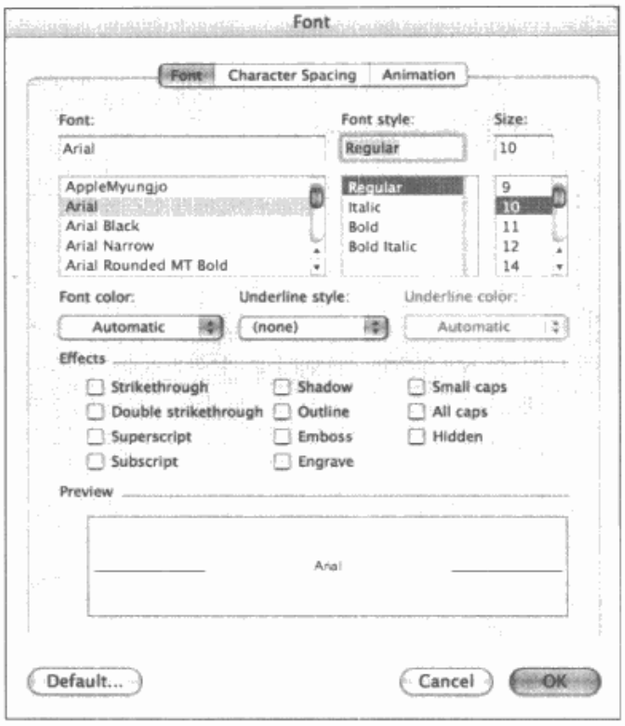


图 4-54 可以使用分类树的例子

- 高度（height）：正常、下标、上标。
- 大写字母（caps）：小型大写字母、全部大写字母。

因此，我们将使用分类树技术去测试选项间的相互影响。

如果您要使用分类树技术，那么有必要准备一个工具。幸运的是，目前在这个网址上可以得到一个免费的工具：www.systematic-testing.com/download/cte_download.php

网站上说这个工具是“临时”免费的，不过实际上已经免费好几年了。下载并安装这个工具，然后花点时间看看用户手册，学习如何使用该工具。

在您使用等价类技术之前，您首先要找出测试问题所在。也就是说您认为可能存在值得关注的相互影响的事物，并应该进行组合测试之所在。

在您找到可能有潜在问题的领域之后；识别出其中的分类，即我在结对测试中提到的“因素”。这些“因素”就是您认为可能会互相影响的事物。在某些情况下，您可能要识别出一个或多个分类的子分类。一旦完成这一步之后，就可以对每一个分类进行标准的等价类划分和边界值分析。得出的结果就是我在结对测试中提到的“选项”。

接下来您要定义分类组合的规则。您想要所有因素都成对组合？某些因素要 3 个一组？有没有例外，例如，哪些类不能组合在一起？一旦完成这个之后，您就可以设计测试用例了。如我们在正交阵列中所做的那样，最终的结果将是一个表。

图 4-55 中是我用 CTE XL 工具画的用于描述字体属性之间相互影响的问题的分类树。您可以看到字体是这棵树的根。大小、类型、删除线、高度、大写字母是这棵树的分类。每个分类有至少 3 个（子）类。

图 4-56 中显示的表格是所有因素都成对组合的结果。表中共有 14 行。第一行表示我们应该测试的字体为：小（8 点）、类型为 regular、双删除线、高度为下标并且全部

pin51.com 最新最全源代码 拼吾爱

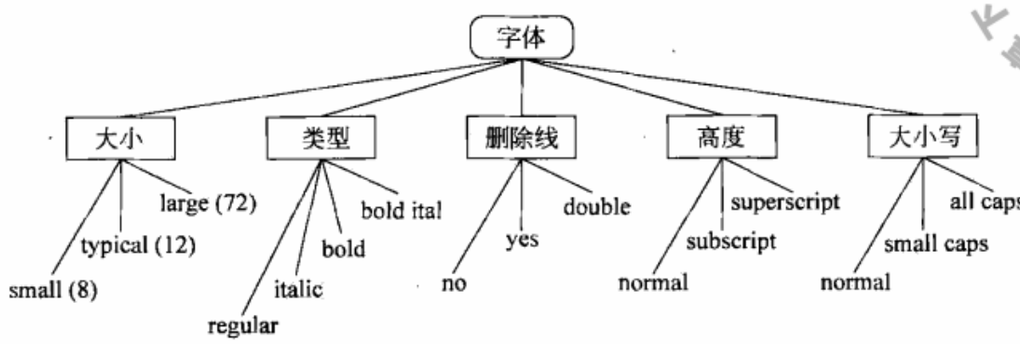


图 4-55 字体分类树示例

大写字母。

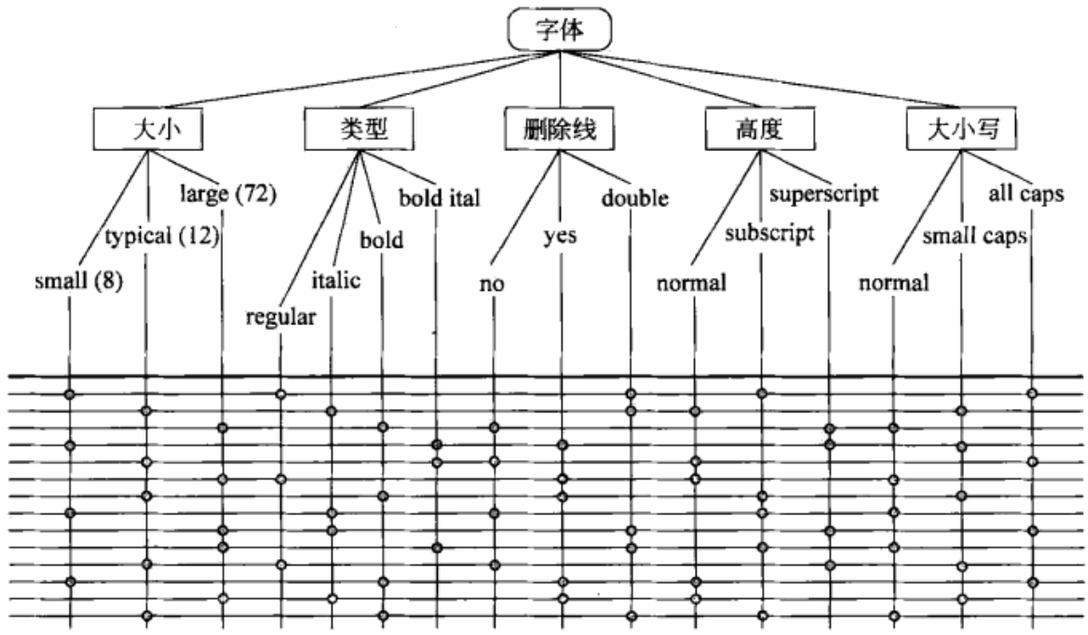


图 4-56 字体成对分类表

到目前为止，分类树能做到的事情结对测试都能做到。但是，在图 4-57 中，您将看到结对测试所做不到的事情。我要求工具为所有因素生成结对测试，并且对大小、高度和大写字母进行三元组测试。表中共有 30 个测试。花点时间研究这个图，并从表中挑选一些组合理解其含义。[6]

4.2.32 分类树练习

在该练习中，根据决策表练习中的背景给出解决方案。

该练习包括 4 个部分：

1. 为决策表创建一棵分类树，列出各个分类以及其等价类。
2. 用工具生成年龄和体质指数的结对测试。

6 如果想要获得更多关于分类树的信息，包括 CTE XL 工具中有什么内容，可以参考首先提出分树概念的文章，马蒂亚-戈洛西曼（Matthias Grochtmann）的 Test case design using Classification Trees（在互联网上搜索一下就能找到），或者可以阅读库曼（Koomen）等人写的 TMap Next 一书中的数据组合测试部分。

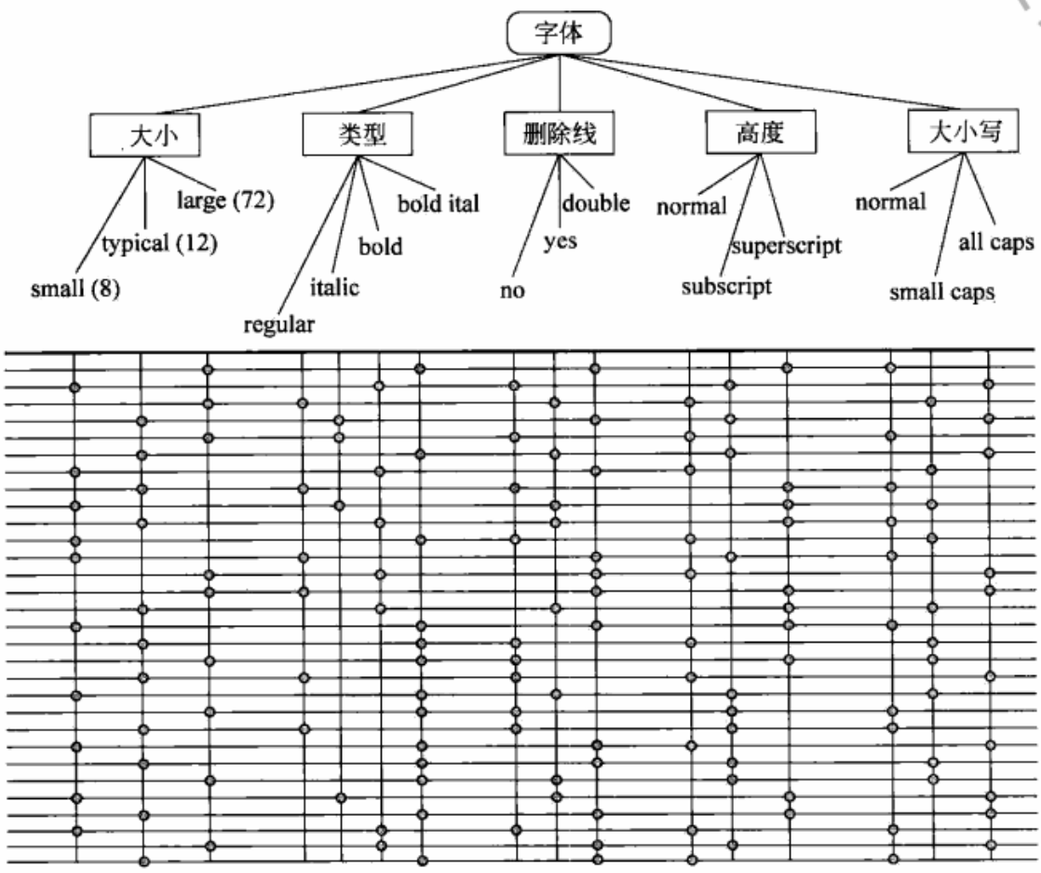


图 4-57 字体三元组分类树

3. 用工具生成所有分类的结对测试。

4. 用工具对所有分类进行三元组测试。

如往常一样，在检查下一步的答案之前，先检查前一部分的答案。

注意：您可能需要下载并安装一个分类树工具。我曾提到过，可以在 www.systematic-testing.com/download/cte_download.php 这个网页中得到一个免费的工具。

4.2.33 分类树练习解答

图 4-58 中显示的是我的分类树，您的树可能与我的有点不同，不过这只是格式问题而已。您的树应该有 6 个分类，其中 4 个分类下是一些布尔类，体质指数和年龄分类下是各个不同的范围，如果不是这样，那么您的分类树就存在问题。

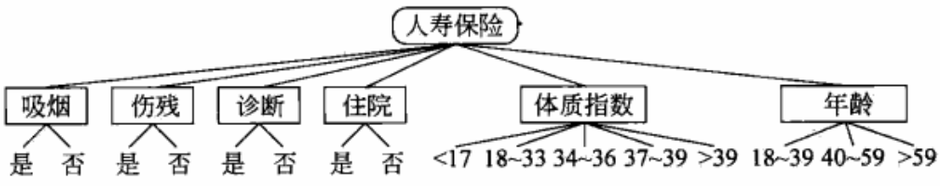


图 4-58 HELLOCARMS 系统中人寿保险的分类树

您可能希望在体质指数和年龄这两个分类下的子分类中列出其边界值，以便进行边界值测试。不过实际上，您可以直接将边界值映射到表格中。在进行成对组合的时候，通常每个选项会有至少两个代表值。

pin51.com 最新最全源码资料下载 拼吾爱

图 4-59 是对体质指数和年龄进行结对测试的分类树和相应的表格。

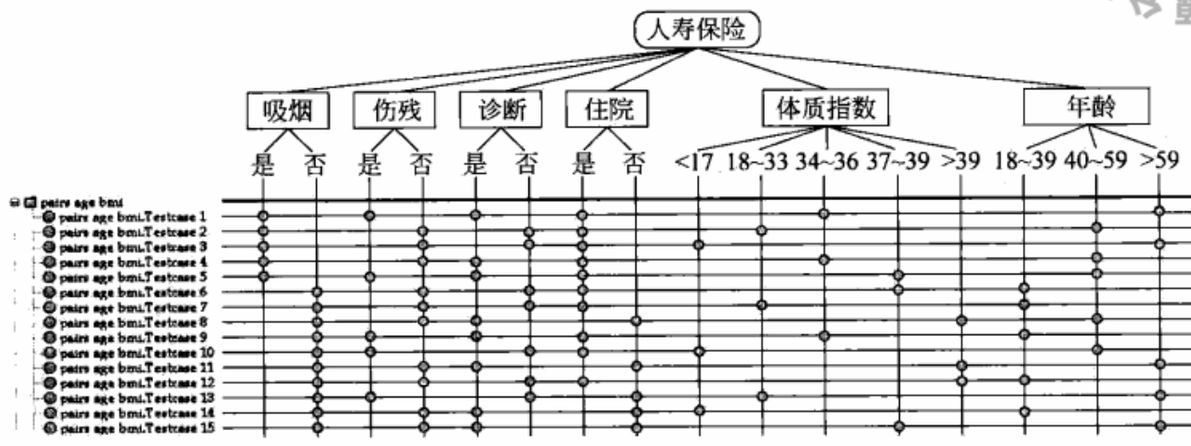


图 4-59 HELLOCARMS 系统分类树测试（体质指数和年龄结对测试）

图 4-60 是对所有分类进行结对测试的分类树和相应的表格。注意，为了方便比较，这个图中保留了图 4-59 中的表格。

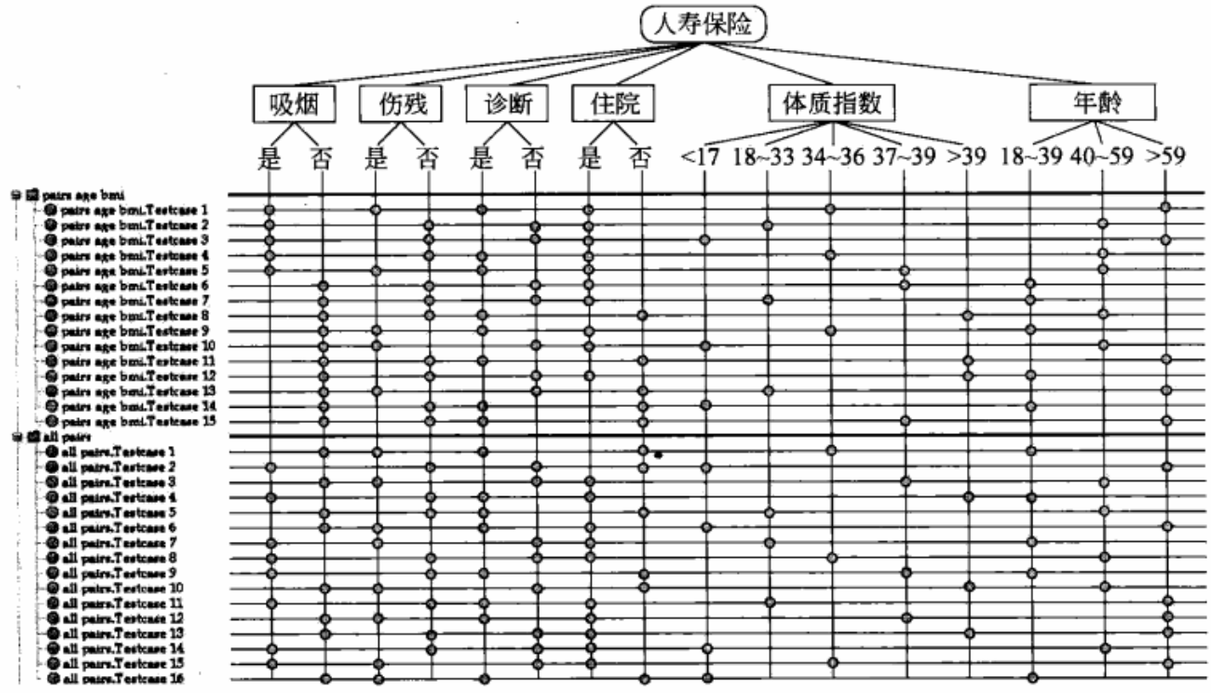


图 4-60 HELLOCARMS 系统分类树测试（所有因素结对测试）

最后，图 4-61 是对所有分类进行三元组测试。注意，为了方便比较，在该图上也包含了前两副图的结果。

4.2.34 根据测试依据设计测试

在结束本节之前，我们将讨论如何将我们所学的测试设计技术与测试依据联系在一起。基于规格说明的测试依赖于测试依据，通常是一份需求规格说明文档。尤其是在高级别的测试中，这一点尤其明显，例如系统测试、系统集成测试和验收测试等。由于需求规格说明主要关注行为，因此我们在本节中讲的这些测试设计技术是行为方面的（或

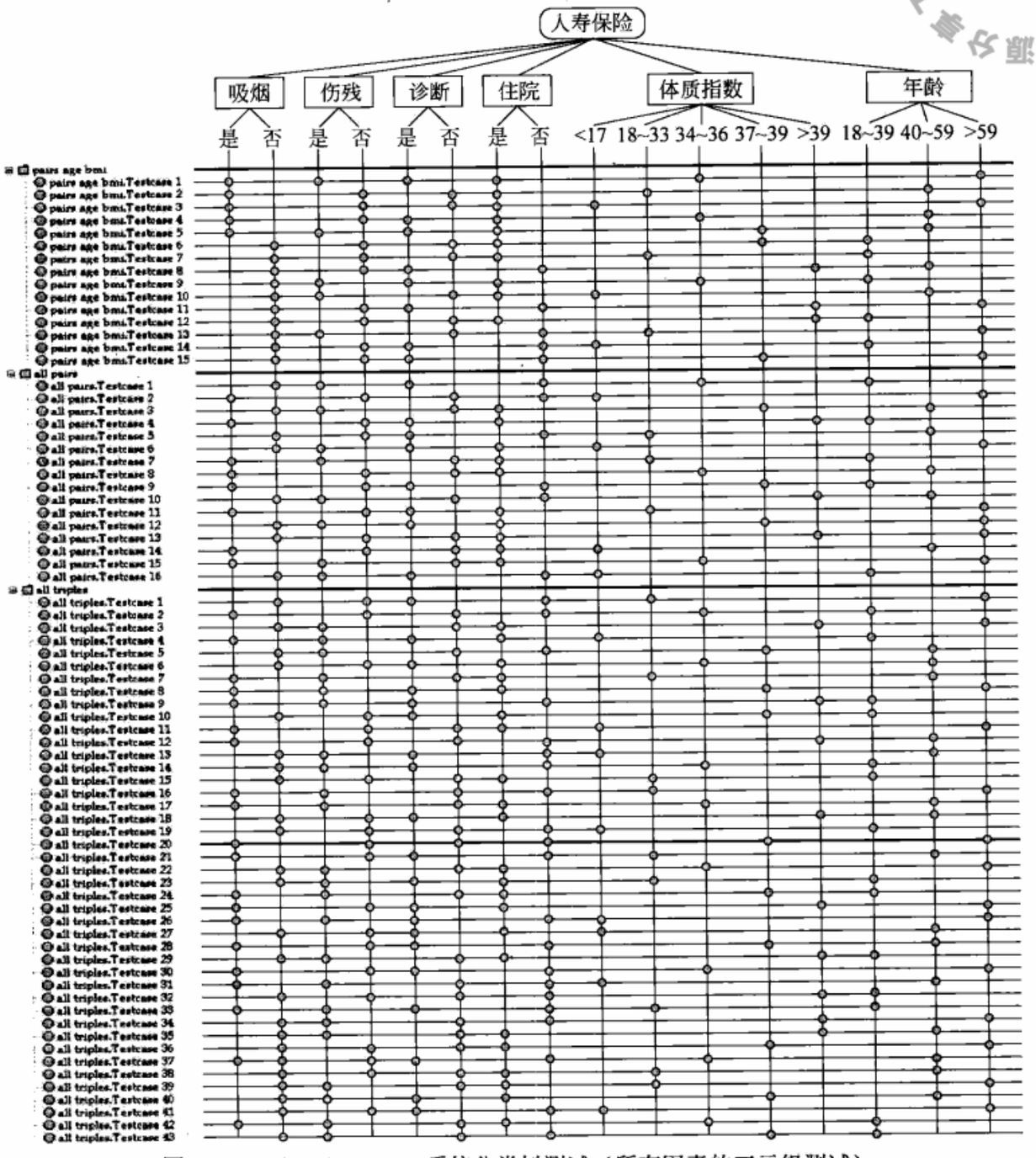


图 4-61 HELLOCARMS 系统分类树测试（所有因素的三元组测试）

黑盒）技术。

一般而言，开发人员、设计人员和业务分析师不会向您提供我们提到过的基本模型。例如，如果您在需求中发现了一个决策表或用例，那太好了，拿来用就可以了。但是，通常您不得不自己想办法去设计这个模型。这里有一个简单的三步过程教您如何使用本节中讲到的技术：

1. 分析需求规格说明。
2. 选择最适合的测试设计技术，并创建相应的基本模型。
3. 根据模型设计测试。

pin51.com 最新最全源码资料下载 拼吾爱

让我们来看一个例子。下面是网络应用程序的两个需求：

- 支持流行的电子邮件附件：TXT，JPEG，GIF，AU，WAV 和 HTML。
- 如果不能识别收到的附件类型，那么自动回复发件人，告诉他哪些附件类型可以发送。

通过与市场人员和设计人员讨论之后，我们发现附件最大为 1 兆字节。

图 4-62 是对上述两个需求进行等价类划分和边界值分析的结果。注意，支持的类型这个等价类，我们将测试 6 种支持的文件类型，同时根据边界值分析，对每种类型进行 5 次测试。对于不支持的类型这个等价类，我们打算测试 4 种不支持的（但典型的）文件类型。我们还将对类型正确，但文件损坏这种情况进行 6 次测试。表 4-16 是根据分析生成的测试用例。您可以花点时间检查一下是否已经覆盖了所有的等价类和边界值。

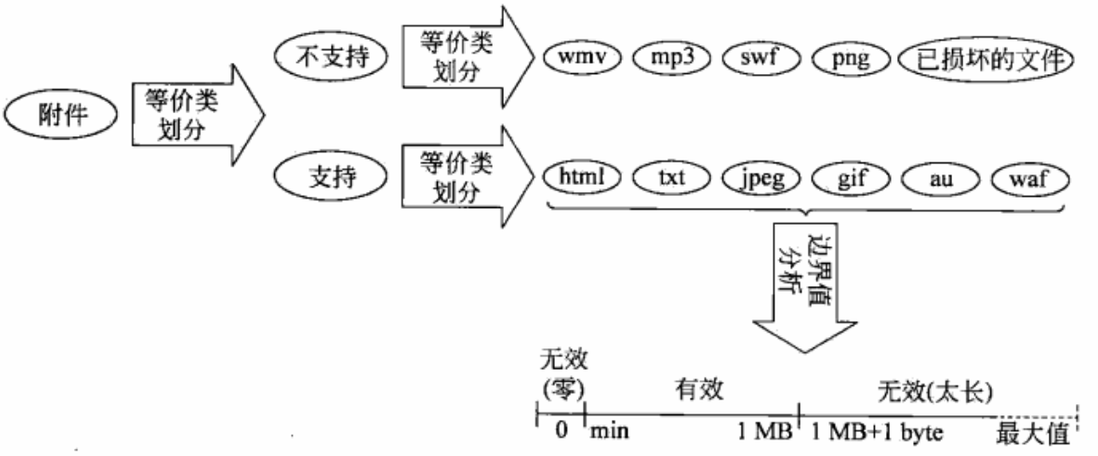


图 4-62　根据需求进行等价类划分和边界值分析示例

表 4-16　等价类和边界值测试示例

#	测试规程步骤	预 期 结 果
1	发送电子邮件，附件为最小的 html	确认能正确接收收件
2	重复第 1 步，但这次附件为 1 兆字节的 html	确认能正确接收收件
3	重复 1、2 步，类型改为 txt、jpeg、gif、au 和 wav	确认能正确接收收件
4	发送大小为 0，html 类型的附件	程序返回错误信息
5	重复第 4 步，类型改为 txt、jpeg、gif、au 和 wav	程序返回错误信息
6	重复 4、5 步，附件大小为 1 兆字节+1 字节	程序向发件人发送错误信息（退件通知）
7	重复第 6 步，所有附件的总大小最大（通常约为 10 兆字节）	程序向发件人发送错误信息（退件通知）。确认未损坏用户的收件箱
8	发送一个大小合法的 wmv 附件	程序向发件人发送错误信息（退件通知）
9	重复第 8 步，改为 mp3、swf 和 png 类型的文件	程序向发件人发送错误信息（退件通知）
10	发送一个大小合法，html 类型但已损坏的附件	程序返回错误信息
11	重复第 10 步，类型改成 txt、jpeg、gif、au 和 wav	程序返回错误信息

pin51.com 最新最全源码分享下载

4.2.35　根据测试依据设计测试的练习

阅读 HELLOCARMS 系统需求文档。考虑需求 010-020-040。

该练习包括两部分：

1. 选择合适的技术进行测试设计。
2. 使用所选择的技术设计测试，并满足该技术所对应的覆盖准则。

如往常一样，在检查第二部分的答案之前，请先看第一部分的答案。

答案在下面。

4.2.36　根据测试依据设计测试的练习解答

首先，回忆一下，在本节以前的练习中，我们曾经测试过电话申请的移交以及拒绝无效房产价值（输入）。在那个练习中，对于贷款申请有如下规则：

- 最小的房产价值是 25 000 美元
- 最大的房产价值是 5 000 000 美元

在此，我们不是要重复这些测试，而是考虑其对费用计算的影响。

我决定使用 3 种测试技术。首先，对费率进行简单的决策表分析，如表 4-17 所示。

表 4-17　HELLOCARMS 系统房产类型费率

条件	1	2	3	4	5
房产类型	家用	出租	商业	分契式/合作式	未开发
操作					
费率	0	1.5	2.5	3.5	4.5

接下来，对于出租和分契式或合作式类型的房产，进行等价类划分，如表 4-18 所示。

表 4-18　房产类型等价类划分

#	划　分	子划分（如果有）
1	家用	—
2	出租	双层
3	出租	公寓
4	出租	度假屋
5	商业	—
6	分契式/合作式	分契式
7	分契式/合作式	合作式
8	未开发	—

我对各种费率进行了边界值分析，包括费率为 0，如表 4-19 所示。您可能认为只要计算零费用、最小费用和最大费用就足够了，但我认为这是一个高风险的功能，需要大量测试。

pin51.com 最新最全软件测试资料 拼吾爱

表 4-19 房产价值边界值

#	划 分	边 界 值
1	0	0
2	0	0
3	1.5	375
4	1.5	75 000
5	2.5	625
6	2.5	125 000
7	3.5	875
8	3.5	175 000
9	4.5	1 125
10	4.5	225 000

使用上述 3 种测试设计技术设计的测试用例如表 4-20 所示。

表 4-20 房产类型和费用测试用例

	测试用例										
条件	1	2	3	4	5	6	7	8	9	10	11
类型	家用	家用	出租	出租	出租	商业	商业	分契/合作式	分契/合作式	未开发	未开发
子类型	—	—	双层	公寓	度假屋	—	—	分契式	合作式	—	—
价值	25K	5M	25K	5M	1M	25K	5M	25K	5M	25K	5M
操作											
费用%	0	0	1.5	1.5	1.5	2.5	2.5	3.5	3.5	4.5	4.5
费用数额	0	0	375	75K	15K	625	125K	875	175K	1,125	225K

注意，我在表 4-21 和表 4-22 中对等价类划分和边界值分析进行了覆盖分析。决策表中的每一列至少有一个等价类和至少两个边界值。因此，只要我覆盖了等价类和边界值，那么肯定也覆盖了决策表。

表 4-21 等价类划分覆盖

#	划 分	子划分（如果有）	测试用例
1	家用	—	1、2
2	出租	双层	3
3	出租	公寓	4
4	出租	度假屋	5
5	商业	—	6、7
6	分契式/合作式	分契式	8
7	分契式/合作式	合作式	9
8	未开发	—	10、11

pin51.com 考试题库

表 4-22　边界值覆盖

#	划　分	边 界 值	测 试 用 例
1	0	0	1
2	0	0	2
3	1.5	375	3
4	1.5	75 000	4
5	2.5	625	6
6	2.5	125 000	7
7	3.5	875	8
8	3.5	175 000	9
9	4.5	1 125	10
10	4.5	225 000	11

4.3　基于结构的测试技术

学习目标：

能回想起该部分内容即可。

本节中的概念主要适用于测试技术分析师，对于测试分析师，本节没有定义学习目标。如果您正在准备相关认证考试，那么您应该看一下高级大纲第 4 章的这一小节，只要熟悉内容并能记住即可。

ISTQB 术语

分支测试：一种白盒测试设计技术，通过设计测试用例来测试分支。

条件测试：一种白盒测试技术，设计测试用例以执行条件的结果。

条件结果：条件判定的结果，为真或假。

条件决定测试（或多重条件测试）：一种白盒测试技术，是对能够独立影响决策结果的单独条件的测试。

控制流：执行组件或系统中的一系列顺序发生的事件或路径。

控制流分析：一种基于控制流图（control flow graph）的静态分析方法。基于事件（或路径）的序列在执行组件或系统过程中进行静态分析。

D-D 路径：未在 ISTQB 术语表中定义。

数据流分析：一种基于变量定义和使用的静态分析模式。

决策测试：白盒测试设计技术的一种，设计测试用例来执行判定结果。

LCSAJ：线性代码序列和跳转。包含以下 3 项（通常通过源代码清单的行号来识别）：可执行语句的线性序列的开始、结束以及在线性序列结尾控制流所转移到的目标行。

pin51.com 最新最全源码 拼吾爱

路径测试：一种白盒测试设计技术，设计的测试用例用于执行路径。

路径：组件/系统从入口到出口的一系列事件（例如，可执行语句）。

语句测试：一种白盒测试设计技术，所设计的测试用例用来执行语句。

基于结构的技术（或白盒测试设计技术）：通过分析组件/系统的内部结构来产生和/或选择测试用例的规程。

4.3.1 基于缺陷和基于经验的技术

学习目标：

（K2）描述使用基于缺陷的测试用例设计技术的原因和原则，区分基于缺陷技术和基于规格说明、基于结构技术之间使用上的区别。

（K2）用例子解释缺陷分类及其应用。

（K2）理解基于经验测试用例设计技术的原则和原因以及何时采用该技术。

（K3）使用探索性测试进行指定、执行和报告测试用例。

（K2）根据缺陷目标，对不同类型的软件缺陷实施攻击所找到的缺陷进行分类。

（K4）分析一个系统，确定为达到特定目标如何选择使用基于详细设计的技术、基于经验的技术或基于缺陷的技术。

我们刚才讨论过了系统化的技术，现在，让我们来看看一些不那么系统化，但是依旧非常有用的测试技术。我们从基于缺陷的和基于缺陷分类的技术开始讲解。

概念上讲，任何时候，只要我们的测试依据是缺陷类型，那么我们就是在进行基于缺陷的测试。通常来说，这种技术的基本模型是过去发现的缺陷的列表。如果这个缺陷列表是以层次结构分类组织的，那么这个测试就是基于缺陷分类的。根据缺陷列表或缺陷分类设计测试用例，就是要设计测试以查找列表中的缺陷是否存在。

ISTQB 术语

基于缺陷的技术（或基于缺陷的测试设计技术）：针对已知的某个种类的缺陷，设计测试用例来测试该类别的缺陷的过程。

缺陷分类法：一种缺陷分类（等级）系统，用于缺陷分类。

基于经验的技术（或基于经验的测试设计技术）：根据测试人员的经验、知识和直觉来进行用例设计和/或选择的一种技术。

对于基于缺陷的测试，我们对覆盖的要求就不那么严格了。大概的准则是我们要为每种缺陷类型设计一个测试用例，但通常是否要设计一个测试用例则取决于其风险等级。换句话说，如果根据缺陷的可能性或影响，证明不值得您花这些工作量去测试这个缺陷，那么就不要做。不过，如果缺陷的可能性或影响的级别比较高，那么您就不能只设计一个测试用例，而是很多。听起来好像很熟悉，是么？

该技术基本的缺陷假设是程序员会重复犯同样的错误。也就是说，一个程序员团队会在各个项目的相似部分犯相似类型的错误。根据这个假设，我们就能基于缺陷的可能性和影响分配测试设计和执行的工作量。

4.3.2 缺陷分类

图 4-63 中是一个缺陷分类的示例。共有 8 个主要类型以及 5 个“簿记管理”类型，这些类型能够帮助我们在缺陷跟踪系统中对缺陷进行分类。接下来，让我们仔细看看这些分类。[7]

- 功能缺陷
 - -规格说明
 - -功能
 - -测试
- 系统缺陷
 - -内部接口
 - -硬件设备
 - -操作系统
 - -软件架构
 - -资源管理
- 进程缺陷
 - -计算
 - -初始化
 - -控制和序列
 - -静态逻辑
 - -其他
- 数据缺陷
 - -类型
 - -结构
 - -初始值
 - -其他
- 代码
- 文档
- 标准
- 其他
- 重复
- 不是问题
- 劣质单元
- 根本原因不明
- 未知

图 4-63 根本原因分类

在功能缺陷这个类型中，有 3 个子类型：

- **规格说明**：功能规格说明（也许是需求文档或其他文档）有错。
- **功能**：规格说明正确，但是具体实现有错。
- **测试**：根据仔细研究，我们在测试数据、测试设计、测试规格说明或其他地方发现了某个问题。

在系统缺陷这个类型中，有 5 个子类型。

- **内部接口**：系统内部交互失败。也就是说，系统内部的某个模块与测试对象之间的集成出现了问题。
- **硬件设备**：系统的硬件或安装系统的主机失效了。
- **操作系统**：操作系统（估计在测试对象的外部）失效了。
- **软件架构**：一些基本的设计假设经证明是无效的，例如假设能在固定的时间段将数据从一个表移到另一个表或者从网络的一端移到另一端。
- **资源管理**：设计假设是可行的，但假设的实现错了。例如数据表的设计不佳而导

7 您可以在我的书 Managing the Testing Process 中找到这个分类，不过我也是参考伯利兹-贝哲（Boris Beizer）的 Software Testing Technique 一书中的分类。

pin51.com 最新最全源码资源分享下载 拼吾爱

致延迟。

在进程缺陷这个类型中，有5个子类型。

- **计算**：软件未能正确处理数学问题。这里不仅仅是指基本的数值计算，也包括复杂的会计或数值分析功能，以及与取整和精度相关的问题。
- **初始化**：当列表中没有数据，程序未初始化等情况下，第一次操作就失败了。
- **控制或序列**：操作的时间不正确或目的错误，例如，浏览界面或域的顺序不正确。
- **静态逻辑**：边界值不正确，等价类未能正确划分等。
- **其他**：不属于上述类型的控制流或处理方面的错误。

在数据缺陷这个类型中，有4个子类型。

- **类型**：使用了错误的数据类型，不管是内建的或是用户自定义的类型。
- **结构**：无效的复杂数据结构或类型，或者使用方式不正确。
- **初始值**：一个数据元素的初始值不正确，例如一个购物数量列表的初始值是零而不是一。
- **其他**：不属于上述类型的数据方面的错误。

代码缺陷类型是指会导致系统失效的代码错误，例如简单的打字错误、拼写错误、格式错误等。理论上而言，编译器会捕获这些错误，但由于目前的浏览器能解析脚本，所以这些错误还是有可能发生的。

文档缺陷类型是指文档中说在Y条件下，系统会执行X操作，但实际上系统执行了Z操作（有效并且正确的操作）的这种情况。

标准缺陷类型是指系统未能满足行业、政府或供应商的标准或规定；未能满足编码或用户界面方面的标准；未能遵守命名规则的情况。

其他缺陷类型是指缺陷的根本原因已知，但又不能归类为上面我们讲过的任何一种类型，如果您的缺陷分类正确，那么其他类型的缺陷应该是很少的。

5个簿记管理类型如下。

- **重复**：两个缺陷报告描述的是同一个缺陷。
- **不是问题**：测试人员提到的系统行为是正确的。这个缺陷报告是由于测试人员误解了系统的正确行为而导致的。这种情况并不属于测试失效，因为这是在测试执行过程中发生的，并且这只是对于实际结果的理解问题。
- **劣质单元**：缺陷是存在的，但这是由于一个随机的硬件失效而导致的，且这个问题在实际生产环境中不太可能遇到。
- **根本原因不明**：测试人员确认修复并关闭了这个缺陷，但是没有开发人员能给出缺陷的根本原因。
- **未知**：没有人知道问题出在哪里。理想情况下，只有很少的缺陷报告会属于这个类型，通常当一个间歇性的缺陷消失了较长的一段时间后，我们就可以认为某些其他的改动修复了这个缺陷。

注意，这里的分类关注的是缺陷的根本原因，至少，从某种意义上来说，最终证明是代码中的错误导致了失效。

您也可以使用一个“过程根本原因”分类，表示这个缺陷是在开发过程的哪个阶段

引入的。注意，我们可能无法使用这样一个分类去设计测试。从测试设计的目的来讲，即使是面向根本原因分类的方法也很难使用。

图 4-64 中是一个粗糙的缺陷分类的示例，源自于本书中我们曾数次提到过的 Internet 应用程序的案例。这个分类关注的是缺陷的症状，这可能更便于我们设计测试。但是，由于这个分类是粗粒度的，所以我们需要额外的信息去设计测试用例，如期望功能的列表、可用性和用户界面的标准和需求规格说明、可靠性规格说明等。在图中，同时还给出了系统测试时各个分类的缺陷百分比。

描　　述	个　　数	%
功能无效	425	46%
缺少功能	179	19%
可用性差	106	11%
构建失败	62	7%
糟糕的系统设计/架构	51	5%
可靠性问题	49	5%
丢失数据	18	2%
糟糕的性能	16	2%
代码过时	16	2%
与规格说明不一致	8	1%
糟糕的用户文档	2	0%
总计	932	100%

图 4-64　根据缺陷症状分类

4.3.3　错误推测

格伦福德-迈尔斯（Glenford Myers）第一个定义了错误推测这个词[8]。错误推测是指测试人员推测程序员可能犯的错误，并根据推测设计测试用例。注意，这就是所谓的一个“灰盒”测试，因为这种技术要求测试人员了解典型的编程错误、错误如何变成缺陷、缺陷又如何变成失效以及我们应该如何触发失效等。

ISTQB 术语

错误推测：根据测试人员以往的经验，猜测在组件或系统中可能出现的缺陷以及错误，并以此为依据来进行特殊的用例设计以暴露这些缺陷。

如果您是根据一个层次结构分类进行错误推测的，例如基于缺陷分类的测试，那么这个分类就是您的模型。若使用了这个分类，那么您还要满足其覆盖准则，要对分类中

8　您可以在第一本关于软件测试的书 The Art of Software Testing 中，找到对该技术的讲解，同时这本书中还讲解了我们曾讨论过的其他技术。

pin51.com 最新最全 拼吾爱

的各个元素进行适当的扩展性测试。不过通常，错误推测大多依赖于测试人员的灵感。

您应该已经知道，测试用例的设计依赖于测试人员的直觉以及对在正确操作系统的情况下，错误（程序员或设计人员的错误）如何变成缺陷并导致失效的过程的了解。您可以将错误推测测试作为一个分析性的、预先设计的、脚本化的测试规程的一部分。不过，通常情况下，测试人员会在测试执行的时候将错误推测测试加入相应脚本（这是比较理想的设计）或直接测试而不进行脚本化（这是相对不太理想的设计）。

错误推测技术的缺陷假设与基于缺陷分类测试的缺陷假设非常相似。然而，在这里我们不仅假设程序员会犯相同的错误，同时我们还假设测试人员了解系统曾经存在过的缺陷并且知道怎么去找到这些缺陷。

此时，您也许会想起我们曾讨论过的质量风险分析。有些概念听起来很相似，不是么？的确，我们在质量风险分析过程中进行错误推测。这是确定风险可能性的很重要的一步。也许我们测试分析师并不会进行错误推测，而是让开发人员为我们做这件事。或者，我们会和开发人员一起做。这取决于我们的经验，这就是为什么错误推测是一种基于经验的测试技术。

也许缺陷分类就是一种风险分析？的确，缺陷分类包含了质量风险分析的一些元素，如潜在的失效列表以及频率等级。不过，通常情况下，缺陷分类中没有缺陷的影响这一元素。

即使缺陷分类包含了潜在缺陷的影响等级这一元素，它依旧不能等同于质量风险分析。记住，我曾说过，质量风险分析是根据质量风险类型或质量特性分类的，而缺陷分类是根据失效类型或缺陷的根本原因分类的。

4.3.4 检验表测试

检验表测试在结构上与质量风险分析非常相似。概念上来讲，就是测试人员将一系列概要的条目列在一张表上，并一一进行检查或提示。有哪些重要的事是系统应该做的？有哪些重要的事是系统不应该做的？

检验表就是测试所使用的基本模型，通常是按主题来组织的，例如质量特性、用户界面标准、关键操作或其他您想测试的主题。在测试执行过程中或在测试设计过程中，如果您要使用检验表测试技术，那么对于每一个测试目标，您都应该设计一个对应的检验表。

检验表测试技术的覆盖准则是检验表中的每个项至少应该有一个测试用例与其对应。然而，检验表项通常是非常概要的。非常概要，并不意味着概要测试用例。您可能记得我们在用例测试的时候谈到过这点。一个概要或逻辑测试用例会告诉您如何生成明确的输入和预期结果，但不会给出确切的值，而那是详细或具体测试用例应该做的。

一个测试检验表比概要测试用例更加抽象。它只告诉您要测试哪些地方、评估哪些特性以及确定测试通过或失败的概要规则等。因此，即使使用同样的检验表，您和其他测试人员设计的测试用例也可能大不相同。注意，我们曾提到过很多基于规格说明的测试技术，而这些技术大多只有一个正确答案。

检验表测试中的基本缺陷假设是检验表中提到的各个区域中的缺陷应该是类似的、

重要的。这一点与基于风险的测试非常相似。但在某些时候，检验表是预先定义好的，而不是通过对系统进行分析而得到的。因此就这点而言，在一个系统化测试策略中，使用检验表测试技术还是占据主导地位的。而当测试使用的是分析性测试策略时，测试条件列表不再是一个静态的检验表，而是通过某种分析，例如质量风险分析，在项目的初始阶段设计好并在项目进行过程中不断更新的。

图 4-65 是一个系统的可用性的检验表示例。让我们来看看这个检验表中提到了哪些系统可用性。[9]

- 简单而自然的对话框
- 使用用户的语言
- 最小化用户的记忆负担
- 一致性
- 反馈信息
- 明确标明退出方式
- 快捷方式
- 恰当的错误消息
- 预防错误
- 帮助和文档

图 4-65 启发式的可用性列表

简单而自然的对话框是指在正确的时间和地点（界面内）向用户提供正确的内容，同时信息量应该是恰当的。界面中的图形或颜色应该能够吸引用户的注意力并增强用户对于信息的理解，而不是与信息相冲突。应该是只显示必要的信息，而不是那些无关紧要的细节。

使用用户的语言是指不要在言语上轻视用户，也不要使用用户不理解的术语、行话以及缩略语。

最小化用户的记忆负担是指从一个界面转换到另一个界面的过程中，不要强迫用户记忆太多的细节，而应该向用户提供一些有用的提示和明显的线索，告诉他们接下去应该做什么或怎么寻求帮助等。

一致性是指界面的风格应该一致、使用同样的快捷键、颜色方案要相似（例如，在整个系统中红表示不合格、绿表示合格）等。

提供反馈信息是指要让用户知道正在发生一些什么事情，尤其是当处理时间正在流逝或某些操作失败的时候。

明确标明退出方法是指要让用户清楚地知道如何退出界面、模型或对话框，退回到原来的地方，包括如何退出系统。

快捷方式是指应该允许专家能跳过新手的引导界面。

恰当的错误消息应该是直观而有用的，而不应该直接使用代码。错误消息应该能准确地描述问题，并且显得礼貌而又抚慰人心。

预防错误是指我们要尽可能防止用户犯错。当然，这样也增加了可测性，因为预防错误意味着我们需要测试的无效测试用例数少了。

虽然除了大多数（技术）支持人员以外，很少有人会使用到帮助文档，但我们还是应该提供这样的文档。

9 我参照了雅各布-尼尔森（Jakob Nielsen）的书 Usability Engineering 中的检验表。

pin51.com 最新最全 拼吾爱

那么，我们应该如何使用这个检验表进行测试呢？对于系统中的所有界面，我们应该系统性地检查每个界面中的主要区域以及每一次界面转换。如果您发现了问题，那么您应该提交一份缺陷报告，描述您在界面上或界面转换过程中遇到的问题。注意，对测试通过或失败的评估也包含了主观的判断。

4.3.5 探索性测试

探索性测试与检验表测试相似，通常需要遵守一些基本的准则，例如一个检验表，而且在评估测试结果的时候也依赖于测试人员的判断和经验。然而，探索性测试本质上更是一种应对性的，更加动态的测试技术，因为探索性测试的大部分操作只有在测试执行的时候才能执行。

概念上而言，探索性测试是一个测试人员在了解系统、设计测试和执行测试的过程中同时进行的。一个测试的结果很大程度上决定了我们下一步要测试什么。

这并不意味着探索性测试是随机的或者由冲动或直觉所驱动的。我们不仅可以将尼尔森（Nielsen）的启发式可用性列表作为一个预定义的检查表，同时也可以通过这个列表了解软件的哪些部分是重要的或者有问题的。完善的探索性测试应该有一个基本模型，无论是记录在纸上的还是存在于记忆中。根据这个模型，我们就可以设计测试用例以对我们感兴趣的地方进行探索。有时候，为了集中注意力同时也方便测试的组织，会在测试章程中限制要测试的区域。

与检验表测试类似，探索性测试的覆盖准则是每一个章程（如果存在）至少应该有一个对应的测试用例。不过，测试章程与检验表项类似，通常是非常概要的。

> **ISTQB 术语**
>
> **探索性测试**：一种非正式的测试设计技术，使用这种技术时，测试人员积极地设计和控制测试的执行，通过在测试中获得的信息来设计新的、更好的测试用例。
>
> **测试章程**：是对测试目标以及应该如何进行测试的可能想法进行说明。在探索性测试中经常使用测试章程。[注：该说明实际上不正确，并不是所有的探索性测试都使用测试章程，测试章程可以应用于所有的动态测试策略。]

基本的缺陷假设是在测试执行过程中系统会暴露出问题区域，而在测试基本分析中是找不到这些问题区域的。也就是说，您只能通过与系统交互才能了解系统行为正确与否。同时这也说明了重要的一点，即探索性测试，或者准确地说是所有基于经验的测试，与其他脚本是互补的，应该很好地融合在一起来使用。

在ISTQB基础级大纲和高级大纲中关注的是分析性测试，即要遵守一个分析性测试策略。分析性测试策略的优点在于预防缺陷、缓解风险以及结构化的覆盖。而基于经验的测试通常遵守动态的或应对性的测试策略。分析性测试策略与动态的策略的测试过程是不同的。

与ISTQB基础级中的测试过程不同，探索性测试的测试过程关注的是我们从系统中探索到了什么信息，根据这些信息我们接下来应该干什么。测试人员并行地了解产品及

其缺陷、计划要进行的测试工作（或根据章程调整计划）、设计并执行测试以及报告测试结果。

探索性测试不能沦落为混乱的、毫无目的地敲击键盘。好的探索性测试应该是有计划的、交互的以及有创造性的。正如我提到过的，测试策略是动态的，例如测试人员在测试执行过程中动态地调整测试目标。[10]

4.3.6 测试章程

由于探索性测试的执行过程中的太多活动都要在测试执行阶段进行，而如果遵照分析性测试策略，许多活动应该在测试执行之前进行。您可以想象，这其中必定面临着一些权衡取舍。其中一点便是对于我们测试了什么、与测试依据有什么关系、以及结果是什么等问题记录地详细程度。

人们经常看到测试人员无法准确而清楚地描述在探索性测试中测试了什么东西，这是一个致命的弱点。如果我们不能说清楚测试了什么、测试了多少以及什么没有测试，那么就意味着我们无法通过测试建立系统能正常运行的信心。注意，分析性测试这方面就做得很好，至少在可追溯的时候。

解决这个问题的一个方法便是使用测试章程。测试章程以很简单的语言描述了任务、目标以及交付物。根据我们为一些客户做的风险分析，我们可以提前设计测试章程，或者我们可以在测试执行之前才开始设计测试章程，并根据测试结果进行不断地更新。

在一个测试的测试执行开始之前就应该根据测试章程计划好探索性测试会话。这里的计划并不要求非常正式，比如遵守 IEEE 829 文档标准或其他什么标准。事实上，可能根本就没有什么纸质的计划书。但是，测试分析师与测试经理之间应该就下列问题达成共识：

- 测试会话要达成什么目标。
- 测试会话关注什么方面。
- 测试会话应该做什么，不应该做什么。
- 需要使用哪些资源，包括要使用多久。

在稍后您会看到测试章程有多简洁，您可能会希望获得更多的信息。的确，在阅读测试章程的时候您还可以参考缺陷分类、检验表、质量风险分析、需求规格说明、用户手册或其他任何有帮助的文档。但是，这些文档只能用作测试执行过程中的参考，而不是作为在测试执行之前的分析对象，例如作为一种测试依据。

在图 4-66 中您可以看到一个测试章程的例子。这是一个真实的探索性测试会话的日志，取自于我们最近为客户做的一个项目。我们负责从用户行为的角度，为开发组织进行验收测试。这个安排听起来有点怪异，不过，这是真的。

在日志的上方，我们可以看到测试员的名字以及测试执行的日期。同时还有测试花费的时间，以及测试人员是否认为这个章程已经有过完整的探索，记住，与检验表一样，这是一种主观的判断。

在日志头部信息的下方就是测试章程。如果有一个测试规程叫做“测试登录安全”、

10 在埃瑞克-范-万尼达尔（Erik van Veenendaal）的 The Testing Practitioner 一书中能找到更多关于探索性测试的内容。

pin51.com 最新最全源码资源下载 拼吾爱

探索性测试日志

测试人员_姓名_____　　日期_什么时候_____
执行时间_1: 45_____　　测试章程是否完成? __yes_____

章程
测试登录页面的安全性，尝试是否能在不输入密码的情况下登录

已报告的缺陷
937－登录框无法抵御SQL注入攻击
939－在密码错误的情况下，系统通过合法用户的登录请求

需要继续跟进的问题
* 3次登录失败后锁定的功能好像不起作用

图 4-66 测试章程示例

或者测试规程中有这么一句话、或者测试设计规格说明中有这样一个条件，都不是什么奇怪事。但是，在这里，可以看到仅仅就这么一句话，没有其他用于测试的细节信息。从这一点您就可以看出探索性测试有多么依赖于测试人员的知识和经验。

在章程的下面是测试的结果，共两个部分。第一部分是缺陷列表。缺陷前面的数字对应着缺陷跟踪系统中的唯一标识。第二部分是需要跟踪的问题。这些问题可能是缺陷，只不过我们没有时间去完成测试。或者也可以表示由于阻塞所以测试未完成。当然，在这种情况中，我们可以认为对章程的探索测试是不完整的。

4.3.7 软件攻击

最后，我们要讲的这种技术您可以认为是所有本节中提到过的基于缺陷的和基于经验的技术的综合体，即软件攻击。它将缺陷分类法、检验表测试、错误推测以及探索性测试等元素结合在了一起。

从概念上而言，您可以将软件攻击视为一种用于迫使系统发生特定失效的、直接而专注的测试技术。也许您会觉得这与缺陷分类法类似，不过软件攻击技术比缺陷分类法更加结构化，因为它是基于一个故障模型的。这个故障模型描述了缺陷是怎么来的，缺陷为什么以及怎样变成系统失效。在稍后讲到缺陷假设的时候，我们会讨论这个问题“缺陷是怎么来的”。

理解缺陷的表现形式是很重要的。仅仅怀疑或认为存在缺陷是不够的。在动态测试中，由于不能接触代码，所以我们只能观察系统的行为。

这就与检验表很相似了。詹姆斯惠特克（James Whittaker）与他在佛罗里达理工（这种技术的发源地）的学生们根据这个故障模型开发了一个简单的攻击列表，用以查找这些故障。这个根据缺陷的产生方式而组织的、层次结构的攻击列表告诉我们如何去触发缺陷，使其表现为系统失效。[11]

11 关于这个技术的所有内容，包括故障模型，在詹姆斯-惠特克 How to Break Software 一书中都有详细描述。

您先要分析如何在被测系统中使用这些特定的攻击方法。然后，再为每一种可用的攻击方法设计相应的测试用例。在这里，测试的分析、设计以及对覆盖的评估是较为随意的，依赖于测试人员的技能、直觉和经验。这种测试技术告诉测试人员什么时候攻击、攻击的是哪些缺陷、如何确定攻击已成功地揭露出缺陷并使其表现为系统失效以及如何进行攻击。不过，两个理性的、有经验的测试人员可能会使用不同的方式对同一个系统进行同一种攻击，并得到不同的结果。

ISTQB 术语

软件攻击（或攻击）：通过使测试对象产生特定类型的失效，有组织、有目的地评估其质量，尤其是可靠性。

那么，在故障模型中如何描述缺陷的来源呢？基本的缺陷假设是缺陷来自于软件与其运行环境的交互以及来自于软件的能力。软件的运行环境包括用户、文件系统、操作系统以及这个环境中其他与被测软件有交互或共存的软件。软件的能力包括接受输入、生成输出、存储数据和执行计算。

在图 4-67 中您可以看到一个被测程序与其运行环境的关系图。程序通过用户界面接受输入并返回输出。它与其他程序通过不同的方式直接交互；例如，通过从一个程序向另一个程序复制粘贴从而进行交互，或通过与另一个程序传输数据并间接地使用数据库管理系统管理数据从而进行交互。

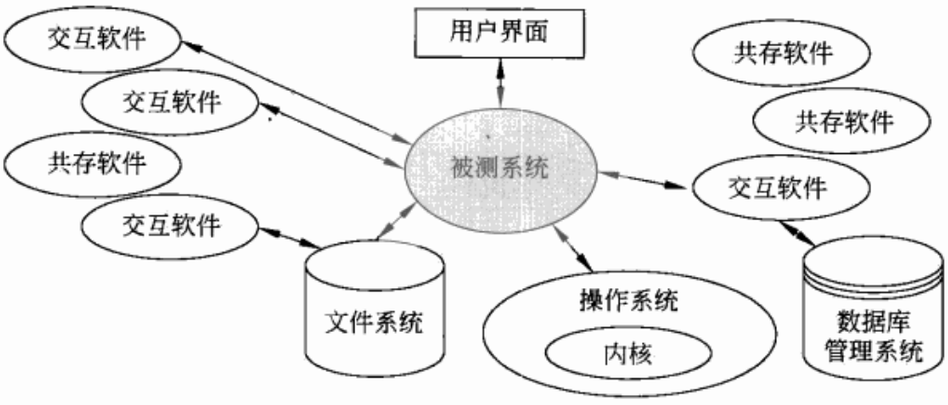


图 4-67　攻击的接口

被测程序也可以通过共享文件数据与其他程序间接交互。还有其他程序，即使并未与被测程序产生交互，但这两者之间还是可能会相互影响，因为，这些程序在同一个系统中运行、共享内存、磁盘、CPU 以及网络资源。

当被测程序创建、更新、读取和删除文件的时候就与文件系统产生了交互。同时被测程序还依赖于操作系统，包括内核和它的库，提供的各种服务以及通过操作系统与硬件间接交互。

那么我们如何攻击这些接口呢？从文件系统开始，有如下攻击方式：

- 填充一个文件，直到最大限度。事实上，您可以边填充边测试，同时观察遇到的

pin51.com 最新最全软件测试工具 拼吾爱

缺陷。

- 与这个相类似的是迫使存储器处于繁忙状态或不可用状态。尤其是当程序同一时间只能做一件事情的时候，例如 DVD 写入程序。
- 您可以临时地或永久地破坏存储媒介。损坏或划花一张 CD。
- 使用无效的文件名，尤其是在名字中使用一些特殊字符。
- 改变一个文件的访问权限，尤其是当这个文件正在使用中或当正处于未使用状态。
- 还有，我喜欢的一种方式，改变或破坏文件内容，稍后我会告诉您为什么我喜欢这种方式。

对于其他共存软件和交互软件的接口以及操作系统接口，有如下攻击方式：

- 尝试触发所有可能的从软件/操作系统接口向应用程序输入的错误。
- 耗尽内存、CPU 和网络资源等相关资源。
- 破坏网络流或内存内数据。

所有上述的这些攻击方式都可能会使用到一些工具，例如自制的、免费软件、或商业软件。

您可能注意到这些攻击技术大多与文件系统相关，但我个人认为这是一种遗憾。我希望能看到更多与系统集成测试、尤其是像交互、端到端测试、数据库之间的数据质量/数据完整性测试等这些测试类型相关的攻击技术。如果是这样，那么接口攻击技术又看起来好像集中于一些基于 PC 的独立软件。

图 4-68 描述了被测程序的能力。应用程序接受输入、执行计算、生成输入。从数据存储中读入数据或向数据存储中写入数据（也许被保存在文件系统中或进程中的数据管理器中，不过我并未在图中画出）。

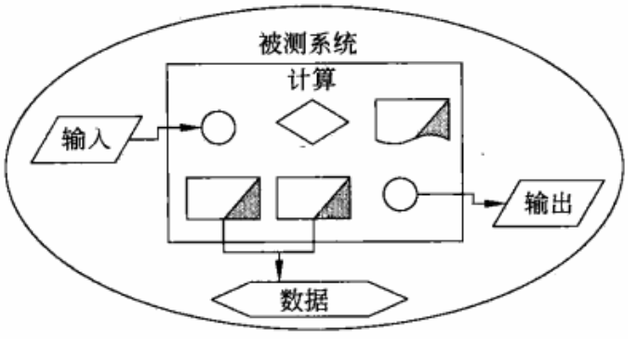


图 4-68　能力攻击

那么，我们应该如何攻击这些接口呢？从输入考虑，我们可以进行以下攻击：

- 我们可以尝试各种输入，以触发所有的错误消息。注意，我们需要一个错误消息列表，这个列表可能来自于：用户手册、需求规格说明、程序员、其他。
- 我们可以让软件使用、生成或恢复默认值。
- 我们可以探索各种许可的输入。在此我们可以使用等价类划分和边界值分析技术。
- 我们可以通过输入非常长的输入值，使缓冲区溢出。

- 我们可以使用会相互影响的输入值，测试各种值的组合，在这过程中可能会用到等价类划分技术和决策表技术（当我们能分析输入值之间的影响时）或结对测试技术和分类树技术（当我们无法分析输入值之间的影响时）。
- 我们可以不断重复相同的输入。

从输出考虑，我们可以进行以下攻击：

- 我们可以尝试迫使软件输出不同的值，包括在输入相同的情况下。
- 我们可以尝试迫使软件输出一些无效值。
- 我们可以迫使软件改变输出值的属性。
- 我们可以迫使软件刷新界面。

问题是这些技术大多是我们只能通过用户界面观测到或者触发攻击。被测软件内到处都可能存在缺陷，作为一个测试人员，我当然会对其他类型的输出、输入、数据和计算感兴趣。的确，这其中某些技术可能与系统接口相关，而这是测试技术分析师的工作，不过我们至少应该对这些技术有所了解。

4.3.8 一个有效攻击示例

也许您能猜到，我经常使用 PowerPoint 程序。有时候使用这个软件是一种冒风险的行为。对此我并不感到惊讶，因为我自己就有这样的经历，PowerPoint 程序与其他的 Office 程序的交互很糟糕，会发生严重的可靠性问题以及数据一致性问题。事实上，在准备课程以及本书材料的过程中，我无意识地对 PowerPoint 进行了两种攻击，给我带来了很大的麻烦。

第一种攻击发生在当我想要在 Word、Excel 以及 PowerPoint 之间共享一些数据的时候。确切地说，我想要在 Word、Excel 以及 PowerPoint 之间复制和粘贴表格。Word 和 Excel 之间能正确地共享表格，但是，PowerPoint 就不行了。事实上，我先得在类似于记事本（去除格式信息）等程序中重新排列表格，然后再一列一列地复制和粘贴。

有时候我会从 Word 中拷贝一些格式化的数据到 PowerPoint 中，尤其是当我将这些文本放到 PowerPoint 的文本框中时，我不经意间触发了第二种攻击，这种攻击使得 PowerPoint 发生了无法恢复的错误。突然之间，在工作过程中，收到一条信息，说您的文件被损坏了，必须退出程序。从上一次存储开始的所有数据都丢失了。更糟糕的是，有时候您已经保存的数据正是引起错误的数据。

为了更好地理解上述这种情况，可以通过修改或毁坏文件的内容（等手段）故意攻击程序来亲自体验。我有一个程序能随机地修改一个文件中的一位或两位数据。如果我对 PowerPoint 的数据文件使用这个修改程序，通常会导致数据文件不可读，不过还是有一些特殊的恢复软件（通常要额外的费用）能恢复所有的文本。

不过，您不必特意使用我这个程序去破坏 PowerPoint 的文件。您只要在 Office 程序之间共享一些数据，一直这么做，直到某一个时刻 PowerPoint 会告诉您文件已损坏。

在这种情况中，PowerPoint 自己制造了无法恢复的错误并导致文件损毁，非常荒谬。您会收到一条消息，说“PowerPoint 发现了一个无法恢复的错误”或者类似的信息。对此我很遗憾，怎么能发生这种事情呢？这是它自己的文件啊。没有其他人在管理这个文

pin51.com 最新最全 拼吾爱

件。没有其他人在向文件内写入数据。如果应用程序自身都不明白自己的文件格式，以及当损坏文件时该如何恢复到正常状态，那么这个应用程序就存在严重的数据质量问题。

4.3.9 其他攻击方式

到目前为止，我已经对高级大纲中提到的攻击技术，即惠特克（Whittaker）技术进行了讲解。然而，您应该意识到，如果您并非在测试独立的、典型的基于 PC 的应用程序的可靠性、互操作性和功能性，那么应该尝试一下其他的攻击类型。

其他类型的软件攻击示例如下，我列出了基本的安全攻击类型：

- 拒绝服务，包括试图向服务器发送大量数据，占用服务器，使其处于不可用状态。
- 分布式的拒绝服务，与拒绝服务攻击类似，使用攻击系统组成的网络进行拒绝服务攻击。
- 我们可以找找有没有后门，即系统中不安全的端口或访问点，或者我们可以尝试自己开一扇后门（通常是通过特洛伊木马程序完成的，例如电子贺卡、电子邮件附件、或色情网站等）。
- 嗅探网络流量，使用嗅探器捕获通过系统某个端口的数据，尝试找出一些敏感数据如密码等。
- 流量欺骗，发送伪装过的 IP 数据包，假冒其他服务器等。
- 电子邮件欺骗，伪装电子邮件的发送地址，这是网络钓鱼攻击的一部分。
- 重放攻击，捕获用户与系统或系统与系统之间的交互，并在稍后进行重放，例如获取账号密码等。
- TCP/IP 劫持，通常是在客户已经登录的情况下，劫持客户与服务器之间的会话。
- 在加密系统中检测弱密钥。
- 密码猜测，可以是逻辑地推测或者根据通用的词典或已知的密码进行暴力破解。
- 病毒，将一段恶意代码附加到一个文件上，或嵌入到一个文件中，执行了这段代码之后，将导致病毒不断复制自己并可能破化系统。
- 蠕虫，与病毒相类似，通常是通过一些安全漏洞入侵被攻击的系统，不断复制自己，同时可能对系统进行破坏。
- 战争拨号器检测不设保护的调制解调器以连接到系统中，不过这种方法已经使用得较少了，不过现在战争拨号器通常会被用于检测不受保护的无线接入点。
- 最后，社会工程是指对使用系统的人进行攻击，而不是系统本身。试图欺骗用户说出他们的密码，修改他们的密码，用电子邮件发送一封文件等。

安全测试，尤其是入侵测试，通常会使用上述攻击方法。有趣的是，在有些用户验收测试中，测试人员要模仿用户，而在安全测试中，测试人员在模仿的“用户”即是黑客。[12]

12 我从格利高里-怀特（Gregory white）等人 Security+ Certification 一书中截取了这个检验表。

pin51.com 软件测试资源分享平台

4.3.10 共同的主题

那么，对于这些基于缺陷的和基于经验的技术，它们还有什么共同点呢？

您可能已经注意到，与基于规格说明的测试技术相比，上述技术的正式程度明显要低很多。另外，这些技术相应的覆盖准则也是非正式的、主观的。

在使用上述技术的时候测试人员必须具备缺陷相关的知识以及其他经验。由于这些技术中的大部分是缺陷聚焦的，因此在查找缺陷方面非常有效。

这些技术是动态的、缺陷聚焦的，而不是分析性的、预防聚焦的。这些技术的动态程度也是变化的。可以是集成在测试执行过程中的快速测试，或者这些技术主导了测试执行过程。在这些测试中，没有正式的预先计划的活动。它们可以包括预先计划的测试会话，同时提供测试章程。它们可以包括创建测试规程文档。

在大多数项目中这些技术都是非常有用的，尤其是在遇到下列情况时：

- 没有可用的规格说明文档。
- 被测系统的文档很差。
- 当您不得不面对一种情况，即在生命周期的早期，用于执行测试过程中的活动的时间不够；尤其是没有足够的时间进行计划、分析、设计、实现。
- 测试人员有相关的经验，如应用领域、基本技术，以及也许是最重要的一点，即测试技术。
- 当我们能分析运行失效的原因，并将这些信息用于缺陷分类、错误推测、检验表、探索性测试以及攻击测试的时候。

我尤其喜欢将基于缺陷的和基于经验的测试技术与基于行为的和基于结构的技术组合在一起使用。在高级大纲中提到的每种技术，无论是测试分析师所用的技术还是测试技术分析师所用的技术，都有其优点和缺点。因此，结构化的技术由于存在一些系统性的缺点所以覆盖不到的地方，使用基于缺陷的或基于经验的测试技术却能覆盖到。下面让我们来看看一个混合使用这些技术的示例，如表 4-23 所示。

表 4-23 选择并混合使用测试技术示例

	脚本化测试	测试章程探索
人员	7 个技术人员	3 个工程师 + 1 经理
经验	< 10 年（总）	> 20 年（总）
测试文档	精确脚本	粗略的测试章程
每天的测试时间（小时）	42	6
发现的缺陷	928（78%）	261（22%）
缺陷有效性	22	44
运行的脚本	850	0
提交的输入	~5 000 – 10 000	~1 000
验证的输出	~4 000 – 8 000	~1 000

在英特网应用程序这个项目中，我们混合使用了动态的、基于测试章程的探索性测

pin51.com 最新最全 拼吾爱

试技术以及分析性的、基于风险的脚本化测试。测试经理以及 3 个测试工程师执行了探索性测试，他们 4 个人共具有超过 20 年的经验。测试技术人员执行了脚本化测试。其中一些是没有测试相关经验的，而有一些人则具备一点点经验。

在测试执行过程中，测试技术人员每人每天大约花六个小时执行测试脚本。一天中其余的 3～4 个小时被用于阅读电子邮件、参加会议、更新缺陷报告、执行确认测试等。测试工程师与测试经理通常有很多其他的工作要做，因此他们每天只能花 1～2 小时用于执行探索性测试。不过您可以看到，即使如此，有经验的测试人员依旧是最有效率的。

不过，就测试覆盖而言，情况就不同了。在 3 个月的系统测试过程中，测试技术人员执行了约 850 个测试脚本。他们的测试覆盖率很高，同时具备良好的文档化的测试结果，因此我们可以向管理层清晰地展示这个结果。而探索性测试并未留下任何明确的文档，我们并未使用我曾提到过的会话日志的方法，部分原因是因为我们依赖于测试技术人员执行脚本测试以取得测试覆盖。

那么，总的输入量呢？我不太确定，不过我估计手动的脚本化测试大概使用了 5000～10 000 个不同类型的输入值，例如字符串、日期、单选按钮等，而探索性测试大概只有其中的 1/5。同样的，与探索性测试相比，脚本化测试用于结果检查的时间要多得多。探索性测试是高效的，不过如果我们要求收集会话日志，那么效率会有所降低。

那么，哪一种技术更好呢？啊，这不是一个比较实验。完全不是一个实验；我只是用这个项目来验证能将两个不同强度的策略混合在一起使用。

探索性测试的查找缺陷的效率非常高，并且我们找到了脚本化测试中无法找到的缺陷。而可重用的测试脚本能有效地缓解风险、回归风险以及建立信心。所以总地来说，这种混合策略是成功的。

4.4 基于缺陷的和基于经验的测试技术练习 1

我们曾讲解过下列技术：

- 等价类划分技术。
- 边界值分析。
- 决策表。
- 用例测试。
- 基于状态的测试。
- 结对测试。
- 分类树。
- 缺陷分类测试。
- 错误推测测试。
- 基于检验表的测试。
- 探索性测试。
- 软件攻击。

在不与先前的练习或示例重复的情况下，指出如何在 HELLOCARMS 系统中使用这

pin51.com 最新最全源码资源下载

些技术。列出相应的程序元素及其描述。

答案在下面。

4.4.1 基于缺陷的和基于经验的测试技术练习解答 1

表 4-24 中列出了我的答案，包括 HELLOCARMS 项目元素以及应该使用的技术。

表 4-24 在 HELLOCARMS 系统中使用所有技术

测 试 技 术	需求片段或元素/描述
等价类划分技术	010-010-040 列出各个域的有效输入，检查每个输入域并确保它们能拒绝无效输入
边界值分析	010-010-040 使用边界值分析对输入确认的测试进行拓展
决策表	010-020-010、010-020-020、010-020-030 根据信贷政策（也许在另一个文档内）设计一个决策表，并根据这个决策表设计测试用例
用例测试	010-010-050 设计一个描述支付功能的用例，并根据这个用例设计测试用例
基于状态的测试	010-010-060 为应用程序开发一个状态转换图（不过不要使用以前练习中电话银行经理的例子）。测试应用程序中基于状态的行为，包括中断并回到面试的能力
结对测试	010-010-220 测试 Globobank 现有产品与潜在客户的成对组合，以及其他银行产品与 Globobank 房屋贷款产品的成对组合
分类树	010-030-040 针对按揭贷款或净值信贷额度支付功能的不同属性设计一棵分类树。设计测试用例覆盖这个分类树，要重点覆盖组合中的边界情况（例如使用整个贷款去偿还已有的一个或多个贷款、偿还现有贷款并刚好能申请新的贷款等）
缺陷分类测试	010-040 根据 Globobank 中所有类似系统，以及其他银行类似系统中找到的与安全相关的失效设计一个缺陷分类表。根据这个分类表设计测试用例
错误推测	整个系统 列出已知的，与 HELLOCARMS 系统存在交互的 LoDoPS、GLADS 以及其他程序中存在的接口问题。设计测试尝试触发这些问题
基于检验表的测试	010-010-020 确保已经测试了所有的界面、界面之间的转换、界面中的信息
探索性测试	010-010-170 将不同的 PC 配置、安全设置、网络连接速度、客户角色以及已有的客户关系组合在一起，用以测试通过网络提交申请
软件攻击	000 介绍 对系统的结构进行攻击，尤其是安全攻击

4.4.2 基于缺陷的和基于经验的测试技术练习 2

在上一个练习中，列出了如何在 HELLOCARMS 系统中使用探索性测试技术或者其

pin51.com 最新最全源码资料下载 拼吾爱

他技术。

本练习包括4个部分：

1. 列出测试该区域所需的一系列测试章程。
2. 您对使用这些测试章程进行测试的测试人员有一些什么要求。
3. 定义每个章程的优先级，并解释为什么这么定义。
4. 说明您将如何报告测试结果。

如往常一样，在查看下一部分的答案之前先检查前一部分的回答。

答案在下面。

4.4.3 基于缺陷的和基于经验的测试技术练习解答2

在上一个练习中，我选择了 010-010-170 需求元素，这个需求要求能通过网络提交申请。我说过我会“将不同的 PC 配置、安全设置、网络连接速度、客户角色以及已有的客户银行关系组合在一起，用以测试通过网络提交申请。”

让我们来看看应该如何做。

首先，我会使用结对测试技术设计一系列目标 PC 的配置组合，包括浏览器类型与版本、操作系统、网络连接速度。我们已经讲过如何使用结对测试技术，所以这里就不再赘述。我会在测试执行之前生成这些配置，并制作硬盘镜像以便在测试中能迅速还原环境。

接下来，我将列出一系列客户角色。角色是指客户的习惯以及他们所拥有的经验。

- 紧张的客户：经常使用返回按钮、修改输入、思考时间比较长。
- 新的网络用户：经常发生数据输入错误、思考时间比较长。
- 熟练用户：快速输入、很少犯错、从其他 PC 程序中复制粘贴数据（例如，账号）、思考时间非常短。
- 没耐心的客户：快速输入、经常犯错、思考时间非常短、多次单击下一步按钮。

下面是各种已有的客户银行关系类型：

- 受限账户，不过都不在 Globobank 中。
- 受限账户，有一些在 Globobank 中。
- 受限账户，所有都在 Globobank 中。
- 拓展账户，不过都不在 Globobank 中。
- 拓展账户，有一些在 Globobank 中。
- 拓展账户，所有都在 Globobank 中。

注意，对于上述两个列表中的内容，测试人员有很大的自主权。同时，与 PC 配置类似，在测试执行开始之前根据已有的客户银行关系类型设计客户数据是有意义的。

最后，我设计的测试章程的列表如表 4-25 所示。

的确，上述章程所覆盖的区域与我们其他基于规格说明的测试所覆盖的区域有点重复。但是，由于测试人员在测试章程中规定的地区之外还会测试其他地方，因此这些场景的实际覆盖要更广。

表 4-25 用于 HELLOCARMS 系统探索性测试的测试章程

测试章程的通用规则：
● 对于每一个章程，在测试之前使用硬盘镜像将您的测试 PC 恢复成未测试过的 PC 配置。确保每种配置至少测试过一次
● 对于每一个章程，选择一个角色。确保每一种角色至少测试过一次
● 对于每一个章程，选择一种已有的客户银行关系类型。确保每一种已有的客户银行关系类至少测试过一次
● 对于每一次贷款申请分配 30～45 分钟的时间；因此每一个章程大约有 30～120 分钟的测试时间
章程：
1 测试两次成功的申请，一次申请是受限的客户银行关系，另一次是拓展的客户银行关系，在这两次申请中客户都拒绝了保险
2 测试一次成功的申请，客户接受了保险
3 测试一次成功的申请，系统拒绝保险
4 测试一次成功的申请，由于房产价值超额而移交申请
5 测试一次成功的申请，由于贷款数目超额而移交申请
6 测试一次不成功的申请，信用历史存在问题
7 测试一次不成功的申请，收入不足
8 测试一次不成功的申请，债务超额
9 测试一次不成功的申请，净值资产不够
10 测试在所有可能的界面尝试取消申请（120 分钟）
11 测试一次欺骗性的申请，用户提供的材料与决策主机中的数据不一致
12 测试一次欺骗性的申请，用户提供的材料与 LoDoPS 系统中的数据不一致

那么，什么样的测试人员可以测试这些章程呢？显然，必须是有经验的测试人员，因为我们只提供了很少的测试规格说明，测试人员的自由度很大。同时由于我不会指导测试人员如何测试这些章程，所以他们必须了解这个系统。他们还得了解一些 PC 技术，至少要知道如何从一个硬盘镜像恢复一种 PC 配置，尽管我可以明确告诉他们应该怎么做。

关于每个章程的优先级，这些章程在表中的先后次序即他们的优先级。注意，我从最简单的用例开始测试，一次没有保险的成功申请，然后再慢慢增加复杂度。我的目标是在执行脚本化测试的过程中并行进行探索性测试。随着脚本化测试的覆盖率越来越高，探索性测试场景的复杂度也越高。

对于结果报告，我会在我的测试管理系统中将每个章程作为一个测试用例进行管理。对于缺陷报告，很重要的一点是测试人员要能够根据隔离法判断问题出在哪里，例如配置、角色、客户银行关系还是系统功能问题。

ISTQB 术语

静态分析：分析软件工件（如：需求或代码），而不执行这些软件产品。

pin51.com 最新最全源码资源下载 拼吾爱

4.5 静 态 分 析

学习目标:

能回想起该部分内容即可

本节中的概念主要适用于测试技术分析师，对于测试分析师，本节没有定义学习目标。如果您正在准备认证考试，那么您应该看一下高级大纲第 4 章的这一小节，只要熟悉内容并能记住即可。

ISTQB 术语

动态分析：组件或系统的执行过程中对其行为评估的过程,例如对内存性能、CPU使用率等的估算。

内存泄漏：程序的动态存储分配逻辑存在的缺陷，导致内存使用完毕后不能收回而不可用，最终导致程序因为内存缺乏而运行失败。

野指针：一个指针，所指的地方在这个指针的范围之外或不存在。

4.6 动 态 分 析

学习目标:

能回想起该部分内容即可

本节中的概念主要适用于测试技术分析师，对于测试分析师，本节没有定义学习目标。如果您正在准备认证考试，那么您应该看一下高级大纲第 4 章的这一小节，只要熟悉内容并能记住即可。

4.7 认证考试模拟试题

在每个章节结束之前，您可以尝试回答以下模拟试题，加强您对所学内容的理解，并准备 ISTQB 高级测试分析师认证考试。

1. 等价类划分技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理条件组合

C 无法正确处理最大值和最小值　　D 无法正确处理某一类输入

2. 边界值分析技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理条件组合

C 无法正确处理最大值和最小值　　D 无法正确处理某一类输入

pin51.com 新客发 新客免费领取大量资源

3. 决策表技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理条件组合

C 无法正确处理最大值和最小值　　D 无法正确处理某一类输入

4. 基于状态的测试技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理条件组合

C 无法正确处理配置组合　　D 无法正确处理某一类输入

5. 分类树测试技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理典型工作流

C 无法正确处理配置组合　　D 无法正确处理某一类输入

6. 用例测试技术能识别的典型缺陷是以下哪一个?

A 无法正确处理最大值和最小值　　B 无法正确处理配置组合

C 无法正确处理某一类输入　　D 无法正确处理典型工作流

7. 结对测试技术能识别的典型缺陷是以下哪一个?

A 无法正确处理事件序列　　B 无法正确处理典型工作流

C 无法正确处理配置组合　　D 无法正确处理某一类输入

8. 假设您是一个银行项目中的测试分析师，该项目要求更新现有的自动取款机系统，允许客户使用该系统支持的信用卡进行借款。对于所有支持的信用卡，系统允许客户借 20～500 美元，包括正好是 20 美元或 500 美元。受支持的信用卡有 American Express、Visa、Japan Credit Card、Eurocard 和 MasterCard。

假设在下列有效测试用例中，三元组中的第一项是信用卡类型、第二项是借款数目、第三项是预期结果。那么下列选项中，哪一个选项的测试用例覆盖了信用卡的等价类划分，并给出了正确的预期结果。

A（American Express，20 美元，成功）;（Visa，100 美元，成功）;（Japan Credit Card，500 美元，成功）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

B（American Express，20 美元，成功）;（Visa，600 美元，失败）;（Japan Credit Card，500 美元，失败）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

C（American Express，20 美元，成功）;（Japan Credit Card，500 美元，成功）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

D（American Express，20 美元，成功）;（Visa，600 美元，成功）;（Japan Credit Card，500 美元，成功）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

9. 假设您是一个银行项目中的测试分析师，该项目要求更新现有的自动取款机系统，允许客户使用该系统支持的信用卡进行借款。对于所有支持的信用卡，系统允许客户借 20～500 美元，包括正好是 20 美元或 500 美元。受支持的信用卡有 American Express、Visa、Japan Credit Card、Eurocard 和 MasterCard。在取款机的用户界面中，默认借款数目是 100 美元，自动取款机有按键允许客户每次增加或减少 20 美元。

假设在下列有效测试用例中，三元组中的第一项是信用卡类型、第二项是借款数目、第三项是预期结果。那么下列选项中，哪一个选项的测试用例覆盖了借款数目的边界

pin51.com 最新最全软件测试工具 拼吾爱

值，并给出了正确的预期结果。

A （American Express，20 美元，成功）;（Visa，500 美元，成功）;（Japan Credit Card，520 美元，失败）;（Eurocard，0 美元，失败）

B （American Express，20 美元，成功）;（Visa，600 美元，失败）;（Japan Credit Card，500 美元，失败）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

C （American Express，20 美元，成功）;（Japan Credit Card，500 美元，成功）;（Eurocard，520 美元，成功）;（MasterCard，400 美元，成功）

D （American Express，20 美元，成功）;（Visa，600 美元，失败）;（Japan Credit Card，500 美元，成功）;（Eurocard，200 美元，成功）;（MasterCard，400 美元，成功）

10. 假设您是一个银行项目中的测试分析师，该项目要求更新现有的自动取款机系统，允许客户使用该系统支持的信用卡进行借款。对于所有支持的信用卡，系统允许客户借 20～500 美元，包括正好是 20 美元或 500 美元。受支持的信用卡有 American Express、Visa、Japan Credit Card、Eurocard 和 MasterCard。在取款机的用户界面中，默认借款数目是 100 美元，自动取款机有按键允许客户每次增加或减少 20 美元。表 4-26 列出的是该系统中交易处理的决策表。

表 4-26 借款系统决策表

条　件	1	2	3	4	5
支持的信用卡	否	是	是	是	是
用户身份验证	—	否	是	是	是
借款在允许范围内	—	—	否	是	是
账户内有余额	—	—	—	否	是
操作					
吐卡	是	是	否	否	否
要求输入新借款数目	否	否	是	是	否
吐钱	否	否	否	否	是

假如您要设计的测试用例满足如下覆盖准则：

- 决策表覆盖。
- 允许的和不允许的借款数目的边界。
- 每一种支持的信用卡成功借款。

设计一组测试用例满足上述覆盖准则，且测试用例的数目要最小。假设每个测试用例只测试一个条件组合和一个操作组合。那么至少需要多少测试用例？

A 4　　B 5　　C 6　　D 10

11. 假设您是一个可编程温控器项目的测试分析师，该温控器用于控制中央加热、湿度、以及空调系统（HVAC）。您要测试这个温控器是否能与中央 HVAC 装置正确交互。在任何时候，HAVC 装置或者处于关闭状态（初始状态），或者处于开启状态。温控器可以向 HVAC 装置发送一个开始事件或一个停止事件。如果这个装置处于开启状态并且它收到了一个停止事件，那么它会停止工作并显示一个“空闲”消息。

如果该装置处于关闭状态并且收到了一个开始事件，那么它将被激活，并且如果所有条件都正常，它还将显示一个“已激活”消息。然而，如果装置处于关闭状态并且收到一个开始事件，那么只要遇到下列一种情况，装置就启动失败：

- HVAC 装置、压缩机、或其他组件没有电源。
- HVAC 装置、压缩机、或其他组件发生故障。
- 防止 HVAC 装置、压缩机、或其他组件发生损坏。

如果启动失败，那么装置会显示一个错误代码以表明是哪种原因导致了失败。

分析上述需求，为该温控器系统画一个状态转换图。在图中，使用不同的转换描述各个造成启动失败的时间/条件对。

下列哪一种描述是正确的？

A 图中有两个状态和 3 个转换　　B 图中有两个状态和 5 个转换

C 图中有 5 个状态和 3 个转换　　D 图中有 5 个状态和 2 个转换

12. 本题背景与上一题相同。根据您画的状态转换图设计测试用例，满足状态和转换覆盖（例如，0-切换覆盖），同时您的测试用例还必须满足下列附加条件：

- 其中一个测试用例必须是从 HVAC 装置的起始状态开始。
- 其中一个测试用例必须是在 HVAC 装置从关闭状态返回到开启状态的时候结束。
- 所有测试用例只能在 HVAC 装置处于关闭状态的时候结束。

测试用例的数目要尽可能少。

您设计了多少测试用例？

A 1　　B 2　　C 4　　D 8

13. 假设您是一个可编程温控器项目的测试分析师，该温控器用于控制中央加热、湿度以及空调系统（HVAC）。除了一般的 HVAC 控制功能外，温控器还能将数据下载到一个基于浏览器的应用程序中，以便进一步分析。

您计划对应用程序和特性进行一些兼容性测试。您所识别的因素以及选项如下所示：

- 支持的 PC/温控器连接：USB 和蓝牙
- 支持的操作系统：Windows 2000、Windows XP、Windows Vista、Mac X、Linux
- 支持的浏览器：Internet Explorer、Firefox、Opera

假设您计划使用分类树方法设计测试用例。由于您担心可能会发生数据传输相关的问题，因此您要覆盖所有操作系统与 PC/温控器连接对，对于浏览器，您只在乎每种类型至少测试过一次即可。

至少要测试多少种配置组合？

A 5　　B 10　　C 15　　D 30

14. 本题与前一题的背景相同。

假设这次您计划使用正交阵列进行测试用例设计。您希望覆盖这 3 个因素的所有选项的成对组合。

至少需要测试多少种配置？

A 5　　B 10　　C 15　　D 30

pin51.com 最新最全源码资料下载 拼吾爱

15. 本题与前一题的背景相同。

假设图 4-69 中是一个非正式的用例，该用例描述了数据传输功能。

温控器数据下载：正常工作流

1. 用户使用 USB 线或蓝牙连接 PC 和温控器
2. 用户通过浏览器访问分析站点（站点地址在用户手册中）
3. 用户单击按钮下载温控器数据
4. 浏览器显示“下载完成”

异常工作流：

- PC 到温控器的连接失败；在步骤 3 后，浏览器显示错误消息“无法连接上温控器”。
- 用户无法访问分析站点；浏览器根据问题的性质显示恰当的错误消息。
- 数据下载半途失败；在步骤 3 后，浏览器显示错误消息“下载失败，请再试一次”。

图 4-69 非正式的数据传输用例

为了覆盖用例中的某一条异常工作流，在测试规程中应该包括下列哪一条描述？

A 单击下载温控器数据按钮，等待“下载完毕”消息，然后进入分析工作流验证数据

B 单击下载温控器数据按钮，然后马上断开 USB 连接或蓝牙连接

C 单击下载温控器数据按钮，等待“下载完毕”消息，然后断开 USB 连接或蓝牙连接

D 单击下载温控器数据按钮，等待“下载完毕”消息，然后关闭浏览器

16. 下列哪一个选项描述了基于规格说明和基于缺陷的测试设计技术的关键不同点？

A 基于规格说明的技术是根据系统的结构设计测试的，而基于缺陷的技术则根据已知的缺陷设计测试

B 基于缺陷的技术是根据系统的结构设计测试的，而基于规格说明的技术则根据系统行为设计测试

C 基于缺陷的技术是根据系统的结构设计测试的，而基于规格说明的技术则根据对测试依据的分析设计测试

D 基于缺陷的技术是根据已知的缺陷设计测试的，而基于规格说明的技术则根据对测试依据的分析设计测试

17. 下列哪一个选项是使用缺陷分类技术进行测试设计的示例？

A 使用缺陷发生的频率作为质量风险分析的输入

B 选择那些可能导致系统失效的输入，而这个失效对应于一个特定的缺陷

C 在测试执行后更新缺陷分类，增加最新发现的缺陷

D 将缺陷分类以及缺陷发生的频率发送给开发部门，进行过程改进

18. 下列哪一项是使用基于经验的测试技术的重要原则？

A 测试技术是测试执行任务安排的关键因素

B 确保 100%的测试人员利用率比测试人员的技术更重要

C 测试人员应该始终关注缺陷预防活动

D 应该根据测试人员找到的缺陷的数目对它们进行评估

19. 假设您是一个可编程温控器项目的测试分析师，该温控器用于控制中央加热、湿度以及空调系统（HVAC）。除了一般的 HVAC 控制功能外，温控器还能将数据下载到一个基于浏览器的应用程序中，以便进一步分析。
考虑下面的这个探索性测试的章程：
从温控器中将不同的数据集下载到应用程序，并对应用程序能够生成的标准报告进行评估。
下面的哪一个选项与高级大纲中对探索性测试方法的描述是一致的？
A 应该在测试执行过程中，且在收到初始的测试对象之后，才执行所有的这些与该章程相关的操作
B 应该在测试执行之前准备好所有的温控器数据集，而所有其他与该章程相关的操作应该在测试执行期间才执行
C 应该在测试执行之前准备好大部分温控器数据集，同时可以在测试执行过程中，根据测试结果修改这些数据集
D 在测试执行开始之前，应该在一个具体测试用例中列出所有与该章程相关的数据、输入和预期结果

20. 下列是惠特克（Whitetaker）提出的软件故障攻击技术：
尝试触发所有可能的从软件/操作系统接口向应用程序输入的错误。
在使用这种技术进行测试的时候，您希望找到下列哪一种故障？
A 当将 Windows 剪贴板中的不支持的字符粘贴入输入域时，应用程序崩溃
B 应用程序启动界面上的公司名拼写错误
C 应用程序无法在报告中用货币格式显示货币数字
D 应用程序无法正确计算信用卡中的总的月欠款余额

21. 假设您是一个银行项目中的测试分析师，该项目要求更新现有的自动取款机系统，允许客户使用该系统支持的信用卡进行借款。
首先用户要插入一张有效类型的信用卡，这时候，系统认为用户处于未验证状态。在借款之前，用户必须进入已验证状态。用户输入正确的密码进行验证。
在验证用户的时候，系统允许用户最多输入 3 次密码，如果第三次密码输入依旧未通过验证，那么系统将吐出信用卡。在第一次和第二次输入失败的时候，系统会提示用户再次输入密码。
如果您担心系统无法正确响应以数字形式输入的无效密码。那么应该使用下列哪一种测试设计技术专门查找这类缺陷？
A 等价类划分技术，辅以边界值分析技术
B 使用正交阵列的结对测试
C 使用 N-1 切换覆盖的基于状态的测试
D 使用二因素验证的分类树测试技术

22. 如果对于程序中的某一特定模块，我们说一组测试满足了该模块 100%的结构化的判定覆盖，这表示？
A 这些测试能找出该模块中所有的缺陷

pin51.com 最新最全源码资料下载 拼吾爱

B 在这些测试中，每一个控制流分支至少执行过一遍

C 在这些测试中，模块中的每一条数据流至少测试过一遍

D 在这些测试中，模块中的每一条路径至少测试过一遍

23. 考虑下面的代码片段：

```
if ((a>b) && (b>c)) {
    b = (a+c)/2;
}
```

假设，在下列选项中，每个括号中的 3 个数字代表了一个测试用例的输入，其中第一个数字代表 *a*，第二个数字代表 *b*，第三个代表 *c*。下面哪一个选项中的测试用例满足了该代码片段的判定覆盖且测试用例数目最少？

A （5，3，2）

B （5，3，2）；（5，4，0）

C （5，4，0）；（4，5，0）

D （4，5，0）；（5，4，5）

24. 下列哪一项是动态分析技术，能用于改善程序性能？

A 代码复杂度分析

B 概况测试（Profiling）

C 网络包嗅探

D 拼写和语法检查

25. 设置-使用对（set-use pair）是下列哪一种静态分析活动的术语？

A 控制流分析

B 数据流分析

C 代码标准分析

D 圈复杂度分析

pin51.com 拼吾爱

第5章 软件特性测试

> “[质量是]一种合用性。产品的特性是决定产品性能和产品满意度的决定性因素……。‘质量’这个词同时还指：没有缺陷……导致投诉、索赔、退货、返工或者其他的损失和问题。这些都是对产品不满意的表现形式。”
>
> Joseph M • Juran 在《质量计划》中指出

高级大纲第 5 章将关注软件特性的测试，如图 5-1 所示。在本章中，高级大纲将对基础级大纲中提到的 ISO 9126 中关于软件质量特性的内容加以展开，并解释测试与各种功能和非功能软件质量属性之间的关系。本章包括以下章节：

1. 概述。
2. 领域测试的质量属性。
3. 技术测试的质量属性。

下面介绍每个章节并讲解它是如何和测试分析关联的。

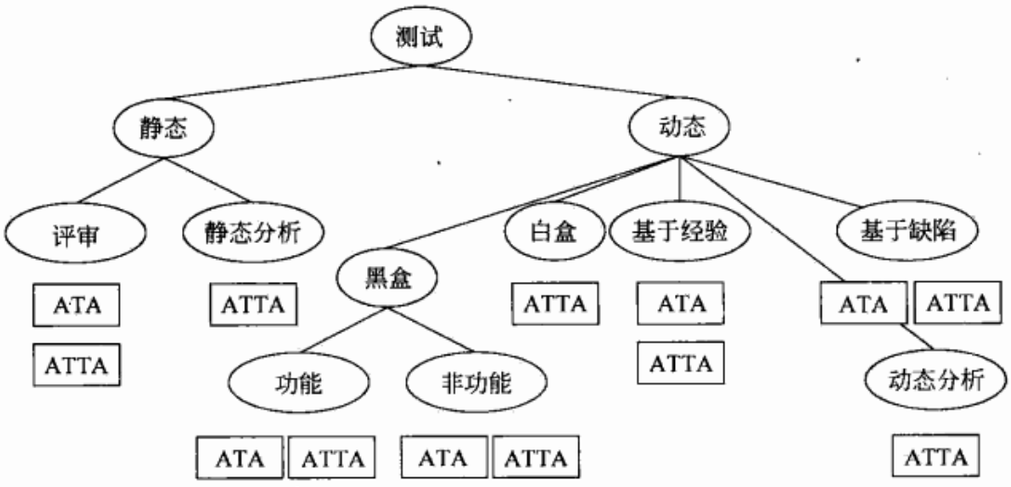


图 5-1　高级大纲中的测试技术

5.1　概　　述

学习目标：

能回想起该部分内容即可。

pin51.com 最新最全 拼吾爱

在第4章的开始部分，讲到了测试的分类，亦即关于测试的一个分类系统。我当时依据ISO 9126标准，阐述了功能和非功能黑盒测试之间的区别。接着我在第4章中讨论了有用的黑盒测试技术，当时并没有再去讨论功能和非功能黑盒测试之间的区别，或是回顾在ISO 9126中定义的质量的特性和子特性等概念。

在本章中，我们将会覆盖这些话题。在这里，我们将考虑如何将第4章中的技术应用到软件应用程序或系统的质量评估中。本书的重点主要是放在测试分析师上，和讲测试技术分析师的姊妹篇侧重点有所不同，而它们共同之处是，我们需要理解质量特性以识别典型的风险，设计恰当的测试策略和确定有效的测试方法。

5.2 领域测试的质量属性

学习目标:

（K4）通过实例，说明在第4章中所列出的哪些测试技术适合对准确性、适应性、互操作性、功能安全性和可达性等特性进行测试的技术。

（K3）使用适当的技术列出、设计、指定和执行易用性测试，实现特定的测试目标和发现特定的缺陷。

功能测试主要关心的是系统做什么而不是系统怎么做。然而，非功能测试则主要关心系统是怎么做它应该完成的工作。因为它们都是关注于系统的行为表现，所以功能测试和非功能测试都是黑盒测试。然而，白盒测试则关注基于系统结构的系统内部是如何工作的。

功能测试通常将需求规格说明作为其测试依据。这应该包括在规格说明文档中描述的需求，还包括那些隐性的需求。测试人员在相关领域的专业知识也可以作为测试依据的一部分。

功能测试会根据测试级别或阶段的不同而有所不同。功能集成测试，主要关注于模块集合体的功能，通常根据部分或全部的用户工作流、用例、用户操作或这些模块提供的特性进行测试。功能系统测试，主要关注于作为整体的应用程序的功能，需要覆盖的用户工作流、用例、操作或特性。功能系统集成测试，将关注在跨越整个集成系统的端到端的功能。

测试分析师可以在功能测试的各个级别，利用各种测试技术进行测试。所有在第4章中讨论过的测试技术都会在某些方面发挥其用处。

我们需要记住的是，测试分析师不是一个头衔、职责描述或仅仅是一个职位，反而，他是一个角色。换言之，有的人专门做测试分析师，而有的人只将其作为他工作的一部分。所以，当专职测试工程师做功能测试时，他们同时无论在职位上还是角色上也是测试分析师；当该领域的专家做与功能测试相关的分析、设计、开发或执行时，他们也是测试分析师；而当开发人员做与功能测试相关的分析、设计、开发或执行时，他们也是

测试分析师。

对于测试分析师来讲，在 ISTQB 高级大纲中我们认为功能测试包含以下几种质量特性：

- 准确性。
- 适应性。
- 互操作性。
- 功能安全性。
- 可达性。
- 可用性。

让我们在本章节接下来的部分中，进一步了解关于每个质量特性的内容。

5.2.1　功能的准确性

功能的准确性测试主要是关心系统是否遵守了已规定的或是隐含的功能需求。换言之，系统是否给出了正确的答案或是正确的响应。在这种情况下，准确性还指测试结果的正确的精确程度。

> **ISTQB 术语**
> **准确性测试：**在 ISTQB 术语表中找不到定义。
> **准确性：**软件产品提供的结果的正确性、一致性或精确程度的能力。

功能的准确性测试还会包括对计算准确性的测试。事实上，如果应用程序是用于实现数学、统计学、会计学、科学、工程学或是其他类似的数学集成功能，对于计算准确性的测试就是至关重要的。功能的准确性测试要求使用很多在第 4 章中提到过的测试技术。

下面来看一个例子。我们将重温之前做过的关于 Quicken 的股票买进/新增窗口的测试设计，尤其是关于股票的数目，每只股票的价格以及授权输入域，如图 5-2 所示。我们应用等价类划分和边界值法分析这些输入域，为每个输入域定义了 13 个特定的输入值。

在这种情况下，我们还想加入合计输入域的测试，该输入域是利用 3 个输入域计算出来的。如我们在图 5-2 中看到的，计算是有问题的，程序没有根据股票的最大数目和每只股票的最高价格的这一组合在合计输入域中给出正确的结果。有可能在系统内部数目是正确的，但是溢出了显示空间。不管怎么样，我都会将此记录为缺陷。

5.2.2　功能的适应性

功能的适应性测试主要关注于系统的一组功能与其计划或指定的任务的适合程度。也就是说，系统是否能够解决我们渴望解决的问题？

pin51.com 最新最全源代码 拼吾爱

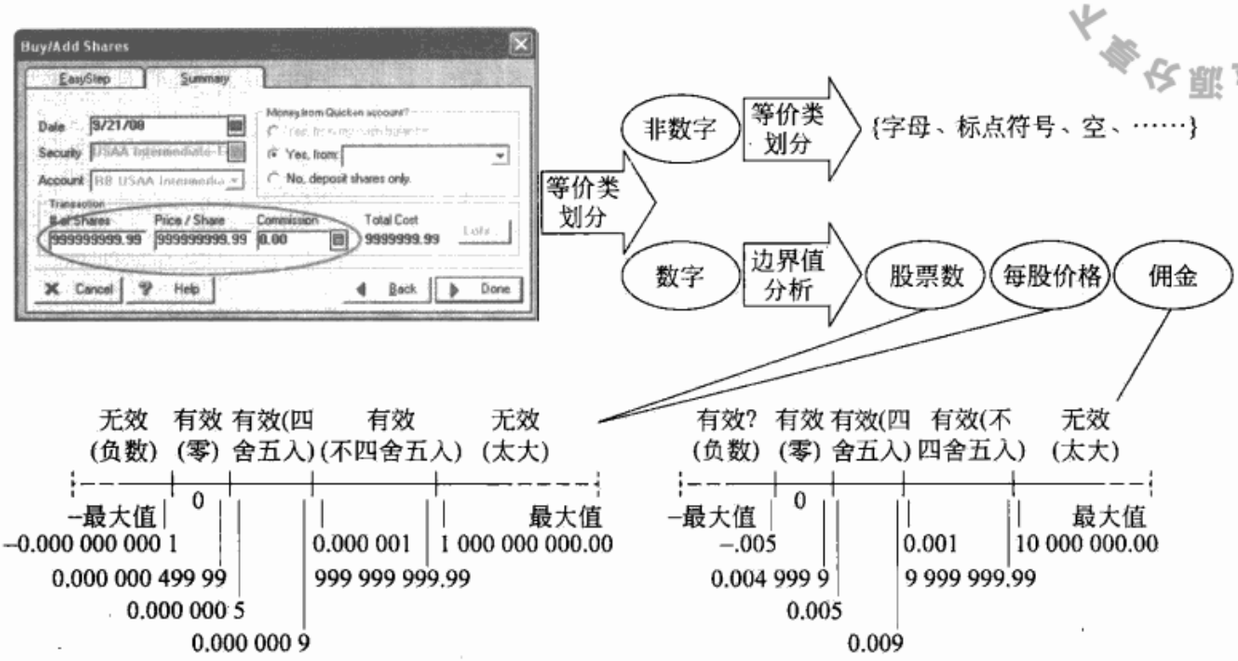


图 5-2 准确性测试举例

在这里我们要关注确认（Validation）这个重要因素。我们打算在一些特定的情况下显示系统的价值。正如在基础大纲中讲过的，已确认的最简洁理解是“我们是否构建了正确的系统？”。而验证（verification）是指遵守了正确的过程，保证功能到需求的追溯性等。检验的最简洁理解是“我们是否正确地构建了系统”。

ISTQB 术语

适应性测试：在 ISTQB 术语表中找不到该定义。

适应性：软件产品为特定任务和用户目标提供一套合适功能的能力。

鉴于我们想要实现的适应性，显然我们需要最大限度地模拟实际的工作流进行测试。为此，我们可以使用用例、测试场景和探索性测试等方法。其他技术，由于过分局限于细节或是太专注于寻找漏洞而偏离了我们的初衷。事实上，您需要很小心地进行探索性测试，以确保测试章程能够反映出现实世界对产品的实际使用需求。

让我们来看一个例子。我们将重温非正式电子商务购物的用例，如图 5-3 所示。总体来说，这个用例讲的是我们应该能将选中的商品放入购物篮中，发起结账的动作，输入购买的相关信息，然后在成交前进行确认。用例还描述了系统需要拒绝空购物篮结账的请求，拒绝购买信息的无效输入，并能识别被放弃的购物篮。

在表 5-1 中，我们通过按照典型的工作流操作测试了系统功能的适应性。我们有一个暗含的需求，就是接受所有 4 种主要的信用卡，所以我们对这 4 种信用卡进行了测试。我们还有个暗含的需求就是支持美国和国际客户，所以我们对这些类型的客户也进行了测试。

在表 5-2 中，我们在一些异常的情况下测试系统的功能。我们检查了当购物篮为空时系统的响应；我们还检查了在每个窗口页面中输入无效信息时系统的响应，包括这之

pin51.com 软件开发资源与考试

电子商务购物：正常的工作流
1. 客户将选中的一件或多件商品放入购物篮中
2. 客户选择结账
3. 系统从客户处收集地址、付款和邮递的相关购物信息
4. 系统显示所有信息，让用户确认
5. 用户向系统确认订单从而发货

例外
- 当购物篮为空时，客户如果进行结账，系统给出提示出错的信息
- 客户给出无效的地址、付款或邮递信息，系统根据具体情况，给出相应的提示出错的信息
- 客户在结账之前或是结账过程中放弃交易；系统在用户闲置 10 分钟之后注销该客户

图 5-3　适应性测试举例：用例

表 5-1　适应性测试举例：（正常情况的）测试

#	测 试 步 骤	预期的结果
1	把一件选中的商品放入购物篮中	购物篮中有被选中的商品
2	单击选择结账	结账窗口
3	输入有效的美国所在的地址、用 American Express 付款、正确的邮递方法及相关信息	每个屏幕显示都是正确的、有效的输入被系统接受
4	检验订单信息	显示与输入一致
5	确认订单	在系统中下订单
6	重复步骤 1～5，但选中两件商品放入购物篮中，用 Visa 付款以及采用国际邮递	如 1～5 中所示
7	重复步骤 1～5，但选中最大件数的商品放入购物篮中，用 MasterCard 付款	如 1～5 中所示
8	重复步骤 1～5，但用 Discover 付款	如 1～5 中所示

表 5-2　适应性测试举例：异常情况的测试

#	测 试 步 骤	预期的结果
1	不要在购物篮中放置任何商品	购物篮是空的
2	单击选择结账	提示错误的信息
3	在购物篮中放入选中的商品，输入无效的地址、付款、邮递的相关信息	提示错误的信息，在信息改正之前无法进入下一个画面
4	检验订单信息	显示与输入一致
5	确认订单	在系统中下订单
6	重复步骤 1～3，但停止活动的状态，在将商品放入购物篮中后放弃交易	在用户处于上次活动状态正好 10 分钟后登出该用户
7	重复步骤 1～3，但停止活动的状态，在每个画面都放弃交易	如 6 中所示
8	重复步骤 1～4，不过不确认订单	如 6 中所示

后，系统是否能阻止我们继续操作直至所有信息被正确填写；最后我们还测试了在所有可能出现的窗口页面中放弃购物的情况。我们无疑保证了从测试到用例的追溯性。

pin51.com 最新最全源分拼吾爱

5.2.3 功能的互操作性

功能的互操作性包括在所有预期的环境下测试功能是否正确。环境当然包括硬件，同时也包括软件、中间设备、链接架构、数据库系统和操作系统。还包括与系统直接产生互操作关系的环境中的要素，与系统间接产生互操作关系的其他要素，甚至是与系统存在简单共栖关系的要素。这种共栖关系是指应用程序之间共同分享同一个网络架构，CPU 容量、内存空间等，但是应用程序之间并不一起工作。

正如您所能想象到的，测试环境存在一个主要的测试配置要素。然后在这些测试环境下测试选定的主要功能。如果您认为某些特殊的功能需要通过一些特殊的方式与特殊的测试环境相互作用，那么一定要确保这种情况被测试到。如果没有这种情况存在，您可以在已确定的测试环境下，测试一些随意的功能组合。

当然，互操作性是指系统之间的相互作用。良好的互操作性指的是在不需要大的调整的情况下就能够容易与其他系统进行集成。设计新功能的时候，是着重考虑（并设计）被测试软件互操作性的重要时机。例如：

- 系统使用业界通用的数据或通讯的标准，例如 XML。
- 系统提供标准、灵活和健壮的接口的能力。

作为一个测试分析师，当您在开发或是整合一些商用现货（COTS）软件和工具时，需要进行大量的互操作性测试。同样的，如果您正在由商用现货或是用户定制的应用程序中衍生出一个综合系统（system of systems），也需要大量的互操作性测试。也就是说，互操作性在进行系统集成测试时是尤为重要的。

> **ISTQB 术语**
> 互操作性测试：判定一个软件产品可交互性的测试过程。
> 互操作性：软件产品与一个或多个指定组件/系统进行交互的能力。

当进行功能的互操作性测试时，尤其是点对点的功能互操作性测试时，您可以使用用例和测试场景进行测试。关于如何定义测试环境：当您理解一个或多个环境、一个或多个功能之间可能存在的相互作用时，您可以使用等价类划分法。当相互作用并不是特别清晰时，可以使用结对测试法和分类树法产生一些相对随意的测试数据组合。

让我们来看一个例子。假设我们将刚刚重温过的关于用例的例子与第 4 章中关于结对测试的例子组合在一起，我们可以使用正交矩阵来组合我们的环境，并以此产生测试用例。

先来看用例，我们有 4 种典型的测试用例：

- 首先是用 American Express，买一件商品，美国国内邮递，我们称之为测试用例 A。
- 其次是用 Visa 卡，买两件商品，国际邮递，我们称之为测试用例 B。
- 再次是用 MasterCard 卡，买购物篮所能装载的最大商品数，美国国内邮递，我们称之为测试用例 C。

■ 最后是用 Discover，买一件商品，美国国内邮递，我们称之为测试用例 D。

由图 5-4 可以看到不同测试环境下有不同的测试用例组合。如果要执行测试，需要先得到正确的配置，然后再运行由用例得到的测试用例。

测试	环境				测试用例
	接入速度	操作系统	安全性	浏览器	
1	DU	XP	Symantec	Firefox	A
2	DU	Vista	Trend	IE	B
3	DU	Mac	McAfee	Opera	C
4	DU	Linux	OS	~	D
5	BB	XP	Trend	Opera	D
6	BB	Vista	Symantec	~	C
7	BB	Mac	OS	Firefox	B
8	BB	Linux	McAfee	IE	A
9	~	XP	McAfee	~	B
10	~	Vista	OS	Opera	A
11	~	Mac	Symantec	IE	D
12	~	Linux	Trend	Firefox	C
13	~	XP	OS	IE	C
14	~	Vista	McAfee	Firefox	D
15	~	Mac	Trend	~	A
16	~	Linux	Symantec	Opera	B

图 5-4　互操作性测试举例

关于异常情况的测试，采用的技术是类似的。请记住我们不希望把异常情况的测试和有效测试混在一起，尤其是，在这里我们测试的是在各种不同的所有支持购买功能的环境配置下，电子商务系统完成购买功能的能力。

5.2.4　功能的互操作性练习

正如在 HELLOCARMS 系统的需求文档中所描述的，HELLOCARMS 系统将与一些其他的系统发生互操作行为。在第 4 章中，您根据用例中典型的或异常的工作流准备了相关的测试。

这里的练习包括以下 3 个部分：

1. 回顾 HELLOCARMS 的需求文档（010 部分），识别 HELLOCARMS 需要与之相互操作的系统。

2. 评估您的用例测试，决定您是否要涉及所有的终端到终端通信。

3. 如果需要的话，建议进行更多的工作，从而保证您的测试可以完整覆盖。

跟平时一样，在进行下一部分的学习之前请先检查自己的答案。

解决方案如下。

pin51.com 最新最全源码资源下载 拼吾爱

5.2.5 功能互操作性练习解答

首先，假设我是在为系统集成测试，而不是组件集成测试或是系统测试。如果这些测试等级您不够清楚的话，应该回顾一下本课程和基础级大纲中与测试级别有关的相关资料。

表 5-3 列出了 HELLOCARMS 系统在其存在的综合系统中与其他系统的互操作性。

表 5-3 HELLOCARMS 的互操作性

系 统	互 操 作 性	需 求 项
GLADS	HELLOCARMS 发送现存的 Globobank 的客户关系信息	010-010-070
GloboRainBQW	GloboRainBQW 发送引导数	003
GloboRainBQW	HELLOCARMS 发送“您是怎么听说我们的”验证码。 GloboRainBQW 发送客户信息	010-030-010
Internet	HELLOCARMS 从互联网用户那里收发申请信息。 互联网用户从 HELLOCARMS 那里收发申请信息	001 003 010-010-170 010-040-040 010-040-050
LoDoPS	HELLOCARMS 发起贷款，并开始办理贷款发放 LoDoPS 发送贷款信息的变更（当提出需求时） LoDoPS 发送更新（当文档已发送） LoDoPS 发送更新（当文档已接收） LoDoPS 发送更新（当材料已发送）	001 003 010-030-050 010-030-120 010-030-140
LoDoPS	HELLOCARMS 发起贷款和着手办理贷款发放 LoDoPS 发送贷款信息的变更（当提出需求时） LoDoPS 发送更新（当文档已发送） LoDoPS 发送更新（当文档已接收） LoDoPS 发送更新（当材料已发送）	001 003 010-030-050 010-030-120 010-030-140 010-030-145
LoDoPS	HELLOCARMS 按照契约规定发送债务信息给退休部门	010-010-050 010-010-100 010-030-040 010-030-070
LoDoPS	HELLOCARMS 将政府退休基金的净收入转换为增加的收入并发送增加的收入信息	010-020-050 010-030-080
LoDoPS	HELLOCARMS 发送所有被拒绝的申请	010-030-130
Scoring Mainframe	HELLOCARMS 发送信息和请求评分（房产的风险、信用评分、贷款与房产价值的比例，以及收入负债比率） Scoring Mainframe 发送分数（房产的风险、信用评分、贷款与房产价值的比例，以及收入负债比率）和最终决定	001 003 010-020-020 010-030-090 010-030-100

续表

系　统	互 操 作 性	需 求 项
Scoring Mainframe	HELLOCARMS 发送联合申请信息和请求的得分 Scoring Mainframe 发送所有请求的得分	010-030-020
Scoring Mainframe	HELLOCARMS 引导 Scoring Mainframe 从联合申请书中删除重复的信用信息 Scoring Mainframe 发送被删除的重复信息的分数	010-030-030
Scoring Mainframe	HELLOCARMS 为申请信息的请求进行排队（如果 Scoring Mainframe 系统不能正常运行）	010-030-110

您可能注意到了我没有去费心列出 HELLOCARMS 内部的联系，例如从费尔班克斯（Fairbanks）的呼叫中心到圣马科斯（San Marcos）的数据中心。这是因为我正在设计关于系统集成的测试而不是组件集成的测试。

关于这点，我需要回顾并扩充之前的表格。首先，我删除了部分指向需求文档的列。其次，我新增加了两个列。第一个列是通讯是否会被基于用例的测试所覆盖，如果这个问题的答案是“是的，肯定会被覆盖”即为“是”；如果这个问题的答案是“不是的，肯定不会被覆盖”即为“否”；如果这个问题的答案是“可能，有可能会被覆盖，取决于测试人员如何运行测试”即为“可能”。

第二个新增的列是测试用例列，描述了它原来的测试用例覆盖了互操作性，还是我建议新增的测试用例覆盖了互操作性。字母 N 表示正常工作流，而字母 E*n* 表示异常工作流，这里 *n* 表示异常的数量。因为测试用例中大部分新增的或是修改的部分变化是很小的，所以您应该回顾并研究一下我的这些测试用例。在很多情况下，新增的部分主要是检查 Scoring Mainframe 或者 LoDoPS 是否发出或收到了正确的信息，如一个关于拒绝的或是取消的申请。

表 5-4 中显示了实际的、可能的和缺失的覆盖率以及关于怎样修正覆盖率中可能或实际存在的空缺的建议。

表 5-4　HELLOCARMS 的互操作性的覆盖率

系　统	互 操 作 性	是否覆盖	用例
GLADS	HELLOCARMS 发送现存的 Globobank 的客户关系信息	否	N
GloboRainBQW	GloboRainBQW 发送引导数	否	N
GloboRainBQW	HELLOCARMS 发送“您是怎么听说我们的”验证码 GloboRainBQW 发送客户信息	否	N
Internet	HELLOCARMS 从互联网用户那里收发申请信息 互联网用户从 HELLOCARMS 那里收发申请信息	否	N
LoDoPS	HELLOCARMS 发起贷款，并开始办理贷款发放 LoDoPS 发送更新（当文档已发送） LoDoPS 发送更新（当文档已接收） LoDoPS 发送更新（当材料已发送）	可能	N E1 E2 E3 E4 E5

pin51.com 最新最全 拼吾爱

续表

系　统	互操作性	是否覆盖	用例
LoDoPS	HELLOCARMS 发起贷款和着手办理贷款发放 LoDoPS 发送贷款信息的变更（当提出需求时） LoDoPS 发送更新（当文档已发送） LoDoPS 发送更新（当文档已接收） LoDoPS 发送更新（当材料已发送）	否	N
LoDoPS	HELLOCARMS 按照契约规定发送债务信息给退休部门	否	N
LoDoPS	HELLOCARMS 将政府退休基金的净收入转换为增加的收入并发送增加的收入信息	否	N
LoDoPS	HELLOCARMS 发送所有被拒绝的申请	可能	N E1 E2 E4 E5
Scoring Mainframe	HELLOCARMS 发送信息和请求评分（房产的风险、信用评分、贷款与房产价值的比例，以及收入负债比率） Scoring Mainframe 发送信用评分（房产的风险、信用评分、贷款与房产价值的比例，以及收入负债比率）和最终决定	可能	N E1 E2 E3 E4 E5
Scoring Mainframe	HELLOCARMS 发送联合申请信息和请求的得分 Scoring Mainframe 发送所有请求的得分	否	N E1 E2
Scoring Mainframe	HELLOCARMS 引导 Scoring Mainframe 从联合申请书中删除重复的信用信息 Scoring Mainframe 发送被删除的重复信息的分数	否	N E1 E2
Scoring Mainframe	HELLOCARMS 为申请信息的请求进行排队（如果 Scoring Mainframe 系统不能正常运行）	否	N E1 E2

5.2.6 功能安全性

功能安全性测试，也被称为渗透测试，主要是用来测试软件能够阻止有意或者无意的非授权访问的能力。企图访问的对象可以是功能也可以是数据，您也可以将这些测试理解为是致力于找出系统安全能力的缺陷。

渗透测试应该包括试图获取您不应该拥有的用户权利；试图获取您不应该看得到的数据；试图获取您不应该拥有的特权，包括删除那些您能看到或是不能看到的数据的这类特权。

理想状况是，在规格说明里就应该有关于安全性信息的说明。然而，现实并非经常如此，很多关于安全性的缺陷已经是众所周知的了，而且更不幸的是，它们还经常反复发生。因此，很多安全性需求在各个系统中都具有普遍性。

渗透测试可以使用更多系统化的技术，但是，就我的经验而言，这很大程度上依赖于类似于攻击和缺陷分类的技术，这些技术在安全性测试中通常被定制使用。

ISTQB 术语

安全性测试：用以决定软件产品安全性的测试。

安全性：软件产品阻止非授权访问能力的属性。这种非授权访问可以是有意或者是无意的，也可以是针对程序或者是针对数据的。

让我们来看一个例子，图 5-5 摘录自一个互联网设备项目的安全性测试计划，该项目我曾经在之前也提到过几次。请阅读该摘录，我将会在接下来的内容中着重指出其中的一些关键要素。

> 该安全性测试工作的目的是：定义在指定日期（MM/DD/YY），XXX 生产环境及客户端中的出现漏洞的基线。测试可以被描述为“以探索和渗透为目的的漏洞基线测试”。所有被认为服务器组群中的组件的服务器以及所有在 XXX 管理控制下或是被 XXX 疏忽的互联网设备都是计划的攻击目标。在系统测试期间的可用客户端将被用来测试漏洞。未被提到的本地机或是设备将不是本项测试的直接目标，但有可能在该测试的外围与本地机发生某种联系。现有的测试中有的可能与安全性测试部分重叠，但是还是由非安全性测试人员来执行。测试工作将在提供给承包商的系统版本上执行，该系统可能需要加载商业和互联网（免费）测试工具。在该安全性测试中，将不会开发自定义软件来模拟轻量级到中量级的攻击的主要威胁。为了保证安全防御的深度或层次，还需要在防火墙的内部和外部进行相关的安全性测试。
>
> 要执行的第二个任务是捕获在 XXX 服务的各个数据流中使用的加密会话。流量的捕获将由商业协议分析软件来执行。该任务主要是为之后的测试提供便利……

图 5-5　渗透测试案例

请注意，在这里我们主要是寻找缺陷，不是为了确认哪些工作正常。结果取决于预算的局限性，还需要注意的是整个服务器环境，被认为是用来当作测试渗透时的当然攻击目标。

在这个项目中，曾经有两个专家顾问运行渗透测试。他们其中的一个写了这份计划，花了一些时间和其他的测试团队成员评估和加强现有的测试，对安全性测试也是很有帮助的。

请注意，其中所提到的威胁的模型，尤其是那些轻量级到中量级的攻击。最后，还需要注意的是一些测试方法是试图获取/回放和中间人攻击，但都是“次要工作”。

5.2.7　可达性

可达性（Accessibility）测试是要测试那些有着特殊需求、受到限制、或是有残疾的人们使用系统的能力。这些需求或者制约等通常产生于国家标准或志愿性指南，有时也会在法律或是合同中做出规定。可达性测试中，针对这些标准或指南，我们通常使用典型的基于需求的测试方法，我们也可能需要使用一些其他的系统化的技术来设计测试。

图 5-6 摘自 U.S. Section 508 法案，这只是若干相类似的要求之一。请注意我们可以使用等价类划分的技术鉴别和测试每个为残障人士设计的辅助技术。顺便提一下，Section 508 法案将“用于残障人士的辅助技术”定义为：“任何通过商业上采购、修改或是定制的用于增加、维护或是改进残障人士的操作能力的技术。它们可以是任何一件物品、设

pin51.com 最新最全源分享下载 拼吾爱

备的一部分、或是一个系统。”

> 当使用电子表格的时候，表格应该允许人们应用为残障人士设计的技术，获取相关信息，域元素以便其实现完成和递交表格的功能，同时还包括辅助残障人士获取所有的说明和提示。

图 5-6 Section 508 节选

现在，您应该想要使用为残障人士设计的技术尝试一些域和功能，但是这存在一定的风险。请记住，我们想要使用基于风险的测试技术，如果被法律或是合同强制要求为每个域和功能提供相关的功能，那么不这么做的影响是什么呢？可能会很严重，所以风险已经高到要测试对每个域和功能的每项为残障人士设计的技术。

ISTQB 术语

可达性测试：测试残障人士使用组件或者系统的容易程度。

5.2.8 可用性

可用性测试，理所当然是关注于用户的。这就是为什么很多著名可用性专家和可用性测试专家大都有心理学的背景，而不是技术或是领域专家。如果拥有社会学和人体工学的相关知识对于可用性测试也是很有帮助的。最后，了解可达性的相关国家标准，对于测试执行这些标准的应用程序也是很重要的。

用户可能因为他们不同的技能、能力和残障而有所不同。有些事情对于一些像我这样的技术老手来说会很容易理解，但可能对我那个只有心理学背景的商业伙伴来说可能就有点困惑。作为一个昔日的 UNIX 开发人员和系统管理人员，我发现其命令行、短命令和互联网基础结构都很容易理解，可是我的商业伙伴有时候却甚至认为简单的多窗口界面太难理解了。

对于科技的使用，儿童显现出超强的智慧。有一次，我将一个带有 CD-ROM 播放器的笔记本电脑放在我 7 岁女儿的房间里，当然我也放了一些 CD 在那里，其中一张 CD 是一本百科全书，当晚，我妻子就听到有一个男人的声音从我女儿的房间里传出来，后来我们才发现，她居然启动了百科全书里的一个设置选项，将文章朗读了出来。

这种“文本转换为语音”和“语音转换为文本”的设置和功能，对于残障人士是非常有帮助的，尤其是那些手的活动受限或是视力损伤的人们。同时，聋哑人或是那些有认知障碍的人，则需要其他类型的为残障人士设计的技术。

ISTQB 术语

可用性测试：用来判定软件产品在规定的情况下，被理解、易学、易操作和吸引用户的程度的测试。

可用性：软件在特定应用条件下，能被理解、学习、使用和吸引用户的能力。

最终，一个可用的软件是适合用户使用的，所以可用性测试是用来度量用户是否认为软件是有效、高效的，他们是否对软件满意。有效性是指软件能够使得用户在预期的使用条件下准确地完全达成他们的目标。高效性是指用户可以在一些符合实际、合乎常理的时间内完成他们要达成的目标。满意度在这里是“令人沮丧”的反义词。换言之就是，一个对产品满意的客户他们可以使用系统有效并高效地达成他们的目标，并认为这个软件如预期的那样对达成这些目标提供帮助。

那么哪些指标会使得用户觉得满意、有效、高效呢？首先是软件的易理解性，也就是使用者是否可以轻松地理解软件能做什么以及为什么需要使用这个软件；另外是软件的易学习性，也就是使用者是否可以轻松使用软件实现其功能；再一个就是软件的易操作性，也就是在软件的功能集中，实现不同特定的难易程度；最后，是软件的吸引力，也就是对用户而言，软件界面讨人喜欢、友好，以及鼓励用户使用。

如果我们正在进行可用性测试，如同大多数其他类型的测试一样，我们可以先设立两类目标，一是检测和消除缺陷，二是验证是否符合之前设定的需求。在可用性测试中，检测和消除缺陷有时被认为是形成性评估，而对需求的测试有时被认为是累积性评估。

在可用性测试中，我们想要观察实际系统在真正的人也就是最终用户那里的效应。（这并不是说测试人员不是真正的人，而是说我们不是那些真正要使用这个系统的人。）要想这样做，我们需要观察用户在实际条件下与系统进行交互，因此有可能会使用到摄影机、模拟的办公室和固定的监测小组等。

可用性测试有时看起来好像只停留在可用性测试这个级别，其实它还可以被整合到功能系统测试中。然而，因为可用性测试相对标准的功能测试而言侧重点有所不同，您可以根据可用性指导方针提高检测一致性，记录可用性缺陷。而这些指导方针应该应用于生命周期的所有阶段，以鼓励开发人员一开始就开发出可用的产品。

ISTQB 术语

启发式评估（Heuristic Evaluation）：一种静态可用性测试技术，判断用户界面和公认的可用性原则的符合度（因此被称为“启发式”）。

可用性测试通常会使用到 3 种主要的技术。第一个是审查（inspection）、评估（evaluation）或评审（review）。这包括从可用性的角度考虑需求规格说明和设计。如所有的这些评审一样，它可以尽早有效和高效地找到缺陷。当您手上已经拿到一些屏幕截图或是测试版本，其实您就可以开始利用真实的用户开展这些工作了。

有一种评审的形式，叫做启发式评估法。它提供了从可用性的角度检查用户界面设计的系统化审查方法。该方法允许我们在设计过程中找出可用性方面的问题，解决这些问题，并在之后进行再次的评估。该过程会一直持续，直到我们从可用性的角度来看对设计已经足够满意了。通常情况下，会选取一个评估员小组，按照公认的可用性准则来对用户界面进行这方面的评估。

可用性测试的第二种形式就是验证实际的执行过程，运行一些可用性测试场景。不同于功能测试场景会关注于输入、输出和结果，可用性测试场景会主要关注可用性方面

pin51.com 最新最全 拼吾爱

的特性，例如学习的速度或可操作性。因为会对用户进行测试前和测试后的会谈，可用性测试场景通常都会超出典型的功能测试场景的范围。在测试前的会谈中，测试人员会得到运行各个部分的指导和操作方法。这些操作方法包括如何运行测试的相关介绍，测试运行的时间，甚至是测试步骤，以及如何记录备注和测试结果，同时也包括即将使用到的访谈和调查的方法。

同时还有语法测试来评估界面，允许用户做什么，不允许用户做什么。还有语义检查，评估信息和输出的意义性。您可以估计到，我们曾经看过的一些黑盒测试技术，包括用例，在这个时候都是有所帮助的。

可用性测试的最后一种形式是调查或问卷调查。这些可以被用以在一个可用性测试实验室中，收集与系统互动的用户行为的观察值。有标准和公开获得的调查。例如软件可用性度量细目表（Software Usability Measurement Inventory，SUMI）和网站分析和度量细目表（Website Analysis & Measurement Inventory，WAMMI）。您可以将一个公用的标准与其他组织和软件进行比较衡量。当然，SUMI 提供的是可用性度量体系，可以用来度量可用性是否完成，是否满足验收标准。

ISTQB 术语

软件可用性度量细目表（SUMI）：一种基于调查表的可用性测试技术，以评估组件/系统的可用性，如用户满意度。

让我们来看一个例子，在图 5-7 中您会看到从一份描述可用性测试场景的文档中节选出来的一些导向性的信息，这是一个我曾经一提再提的关于互联网设备的项目，请注意我们在第一段中定义出了测试的目标。

> 在这个检查清单的每个项目都从属于一个可用性特性或是被测试系统的质量，这些可用性特性和系统质量都是针对一个非常初级的用户为待测试系统中拆包、组装、开机和配置软件的功能有效性的。以下这份检查清单将预估最终用户使用该系统的体会。
>
> 该测试清单包括 4 个主要的场景：
>
> - 打包和硬件（100 分）。
> - 软件安装和配置（100 分）。
> - 互联网连接和在线注册（50 分）。
> - 软件发现和使用（50 分）。
>
> 对于每个部分而言，通过为每个问题选择最合适的答案来完成检查清单。要想为每个部分打分，就要为每个选择的答案进行打分，再在每个部分最后的一览表中记录所有的分数。

图 5-7　可用性测试大纲举例

下面一段描述了测试集的结构，包括 4 个主要的场景，并根据设计者对重要性的理解，为每个场景设置了权重。之后又介绍了一些关于如何使用检查清单的简单的指引。

在图 5-8 中，您会看到测试场景中的指定步骤之一。您可能会猜到，它是来自于打包和硬件场景。这个步骤是用来检查用户是否有一个可行的方法判断在表中的假设是否

与实际相符。执行测试很简单，只需要选中您认为合适的复选框，然后在所有场景都完成之后将分数相加。

（pts）有没有一个部件细目表让客户可以判断是否还存在遗漏？（例如，外箱标签、装箱单，或是建立一个“产品安装路线图”）

5 □ 细目表完整，并包括带有详细信息的图片可以用来准确识别部件
3 □ 细目表完整，但是只有文字说明
2 □ 细目表不完整
0 □ 没有细目表

图 5-8　设计、详细说明和执行的举例

5.2.9　可用性练习

一开始只有 Globobank 的员工使用 HELLOCARMS 系统，但最终总会有一批不同类型的用户像他们一样使用这个系统。因此，对各个类型的用户都要具有可用性就显得尤为重要了。

该练习包括两个部分：

1. 找出至少 3 个 HELLOCARMS 系统的用户群。

2. 对每个用户群，概括出对应易理解性、易学习性、易操作性和有吸引力这些属性，分别需要进行哪些可用性测试。

跟以往一样，在开始下一个部分之前请先检查自己的答案。

答案会在下面给出。

5.2.10　可用性练习解答

该系统的用户群包括：

- 电话银行家和高级电话银行家。
- 客户（通过互联网）。
- 在分行工作的职员。

在 HELLOCARMS 系统的需求文档中还能找出一些其他的用户群。

该练习的下一个部分要求您给出为覆盖可用性 4 个特性而进行的两个或 3 个可用性测试项目。

首先让我们从 ISO 9126 中找出关于可用性测试的每个子属性的定义：

- **易理解性**：软件产品使用户能够理解软件是否适用，以及该软件如何在特定条件下完成某些特定任务的能力。
- **易学习性**：软件产品允许用户学习其应用的能力。
- **易操作性**：软件产品允许用户操作和控制其的能力。
- **吸引力**：软件产品吸引用户使用的能力。

以下是我为各个用户群设计的可用性测试：

电话银行客户经理和高级电话银行客户经理

- 对于易学习性，我需要让一些真正的银行客户经理先通读用户指南的初稿，有可

pin51.com 最新最全 拼吾爱

能也要他们通读相关的用户培训资料（尽管我希望不需要对该应用程序进行培训用户即能使用该软件）。然后，我会调查他们是否发现哪些功能是易学的。当运行一些实际的测试之后（详见下一要点），我会再次调查他们，以了解该用户指南在帮助他们上手使用该系统的过程中起到了多大的作用。

- 对于易理解性和易操作性，我会让一些实际的银行客户经理参与到 alpha 测试中（会与系统集成测试有一定的重合），并会让他们进行一些基于用例的测试，然后调查他们测试和使用的经历。
- 最后，当测试结束之后，我会让用户（银行客户经理）重新看一些系统的截屏，对配色方案、图像和文本从吸引力的角度进行评分。

客户

我将会重复我曾经为电话银行客户经理设置的测试，但是会做出一些改动，我不会在易学习性场景（测试中）使用用户培训或用户指南，而是会将其集成到产品的使用中，要求他们在需要时使用在线帮助，从而在使用的过程中学习如何使用该应用程序。我只会对在测试结束后对用户进行一次调查，了解他们对易学习性的感受如何。

分行职员

我将会重复我曾经为电话银行客户经理设置的测试，但是会做出以下两点改动：

1. 我估计他们会很忙，所以不会提供培训和用户手册。

2. 我将会运行一次 alpha 测试，然后等两周，然后再运行一次。这时，我会调查那些银行职员是否感觉在一段时间不使用该系统之后再次上手会变得困难。

5.3 技术测试的质量属性

学习目标：

（K2）解释在测试策略中包含效率、可靠性、技术安全性测试的理由，举例说明这些测试可以发现的缺陷类型。

（K2）根据典型的缺陷目标对技术测试中的非功能性测试的特性进行描述，说明它们在软件生命周期中的典型应用，以及哪些技术适合测试设计。

大纲中的这部分内容主要是针对测试技术分析师的。对于他们来讲，本章节的学习目标就会更有深度了。然而，作为一名高级测试分析师，很有必要来理解我们的技术同事在做什么以及为什么这样做。我们还应该知道他们倾向于通过这些测试发现哪些缺陷；同时也可以用以提醒自己，单单进行功能测试还是不够的。

我要重申一下之前提过的事情，我们在之前的章节中提到过的功能测试，其侧重点是系统能做什么而不是系统怎么做的。而非功能测试的侧重点便是关注系统如何进行它的工作。无论是功能测试还是非功能测试都是黑盒测试，也就是说关注于系统的行为。然而，白盒测试则更关注于系统内部是如何工作的，也就是说关注于系统的结构。

非功能性测试可以也应该贯穿于软件的生命周期，并且尤其在组件测试和集成测试中可以提供相应的帮助。当性能存在高风险时（通常也确实是这样），性能测试就应该渗

透到测试的各个级别中，包括一些类似于组件测试和集成测试这样的早期的测试级别中。与之相似的是，在许多系统中，尤其是在那些嵌入式系统中，资源的使用是一个值得关注的事项，因此这些测试应该开始得越早越好。

在系统测试和操作验收测试等较高级别的测试中，非功能测试是一个非常重要的部分。这些测试可以覆盖在 ISO 9126 中任何一个非功能性的质量特性和子特性，我们将在本课程中重点讲解其中的一部分内容。遗憾的是，一些类型的非功能测试是非常昂贵的。所以，我们经常需要尝试在有特定目的的硬件和软件的组合中，去测试一些指定的、与客户使用相似的系统。这里定义的硬件和软件的组合包括服务器、客户端、数据库、网络和其他资源。因为只使用了有限的资源，导致这些项目会有一定的局限性，所以在决定这样做之前务必需要考虑相关的风险。

尽管有些工作的所有权还很有可能会有所变化，有些时候非功能性测试也可以在软件仍在开发的时候进行。例如，基于一个软件产品，会有各种各样与可靠性、资源的使用和软件的性能有关的服务水平协议。在这种情况下，监控就是很有必要的了。

在本章节余下的内容中，让我们来看看适用的质量特性，以及一名测试技术分析师是如何测试这些质量特性的。

5.3.1　技术安全性

我们要看的第一个质量特性是安全性。安全性是很多应用程序的一个关键的风险项，而且在最近得到了很多的关注。安全性是软件的是第一个质量属性，而且好像引起了很多立法者的注意，因为现在越来越多保障信息隐私和信息安全的法律陆续出台，而且比起其他的质量特性，更多的软件供应商因为安全性的问题得到法律的制裁。

安全性测试及其失效与功能性测试及其失效是有所不同的。例如，安全性测试通常试图利用一些意想不到的副作用和不好的假设来进行测试，而往往许多缺陷和失效就隐藏在这些副作用和假设当中。大纲中曾经有一个举例，讲的是，一个多媒体播放器正确播放音频的同时会将文件写入非加密的临时存储中。如果您能找到非加密的文件，也就意味着您破坏了所有者的知识产权。

还有另外一个很好的例子是，我们曾经为一个客户做过这样一个项目，我们在系统测试的早期对其各项服务进行安全性测试。我们发现了很多缺陷，然后开发人员进行了修复。之后，就在即将进入生产阶段之际，我们再次进行了安全测试，我们发现在第一次安全性测试搭建环境使用的所有服务器工作得都很正常。但是，所有新增的服务器却在运行过程中出现了第一次测试时出现的同样的问题。

安全漏洞往往与以下行为相关：数据存取，功能使用的特殊权限，以及向系统植入恶意程序，拒绝合法用户使用系统，找出或捕获机密数据，破解加密的交易（例如破解密码和信用卡信息），以及发布病毒等行为。漏洞可能会存在于界面、文件系统、操作系统和外部软件中。

关于安全性有一件有意思的事情，就是安全性的质量越高，可用性、性能和功能性的质量就越低。例如，如果在用户输入错误的数值时，系统能出给一个有帮助的提示，对于用户而言肯定是个不错的功能，但是，我们可以想象一下如果是在密码输入域中，

pin51.com 最新最全 拼吾爱 下载站

这些提示将会成为黑客和其他恶意用户的突破点。

我们在第 4 章中曾简略介绍过安全性攻击的内容，在之前的章节中，我们也讨论过功能安全性测试（也叫做渗透测试）方面的相关内容。接下来，我们会讨论有关如何设计技术安全性测试的几个构想。

一个是信息检索。这里我们用到了一个很有效的、古老的研究方法（检索法）以发现（互联网上）一些有价值的弱点。如果您知道如何正确使用的话，当今这些方法应用在如何有效检索互联网海量数据上其实并不那么陈旧。这里指的些信息可以包括雇员名字、公司或是数据中心的物理地址，以及最好是有互联网和通讯基础设施，端口的 IP 地址，无线接入点和正在使用的软硬件和操作系统的类型和版本。众多的组织都靠“掩盖（事实）而使之安全”，而麻烦在于，现在掩盖（事实）的难度相比过去开始变得越来越高了。

另一个是漏洞扫描。这通常会涉及到使用一些开源的黑客工具，例如 John the Ripper、SATAN、Cain 和 Network Stumbler。这些工具可以帮助您找出安全性的漏洞，或者是可以用来破坏安全性的漏洞。同样现在还有各种漏洞检查表可供参考或采用。信息检索尤其有助于发现漏洞，因为您需要非常清楚在哪里使用这些工具，而检查表能帮助您很好地定位这些漏洞。或者，您可以使用那些除公司 URL 之外包含微量（甚至几乎没有）其他信息的漏洞，这也被称为“零知识”渗透测试。

在使用信息检索，或者漏洞扫描，或者是两者一起使用后，您都可以制定出一个攻击计划，从而概括出您确信能够危及某系统的安全性或某组织安全策略的相关测试活动。攻击需要指定目标接口，或者至少要在该目标接口上运行一些相应的逻辑测试用例。

5.3.2 安全性攻击

这些逻辑测试用例的一个很好的来源就是安全性攻击。我们在本课程的第 4 章中曾看过其中的一个例子，我们还看了如何利用 Whittaker 的“如何破坏软件”模型进行功能攻击测试。Whittaker 还将这一概念扩展至安全性，如有必要可以进一步了解一下。[1]

为了帮助您回忆，在图 5-9 中，您会看到一幅图，描述的是一个应用程序正在其使用环境下进行测试。该应用程序从其用户界面输入和输出信息。它通过各种形式与其他程序进行直接交互，例如，从一个应用程序中复制，粘贴至另外一个应用程序，或者是在数据库管理系统中，向一个应用程序发送数据，再将数据取回，这样进行管理数据。

通过在一个文件里共享数据，以上这个在测试中的应用程序还可以与另外一个应用程序进行间接交互。甚至，有些应用程序就是不与该应用程序进行直接或者间接的交互，也可能对其产生潜在的影响，反之亦然，这是因为事实上它们共同存在于同一个系统中，共享内存、硬盘、CPU 和网络资源。

这个应用程序在新建、更新、读取和删除文件时，都要从文件系统收发数据。它同样依赖于操作系统，无论是资料库还是内核，为它提供与不同的服务和硬件进行交互媒

1 见 James Whittaker 的 How to Break Software Security 一书。

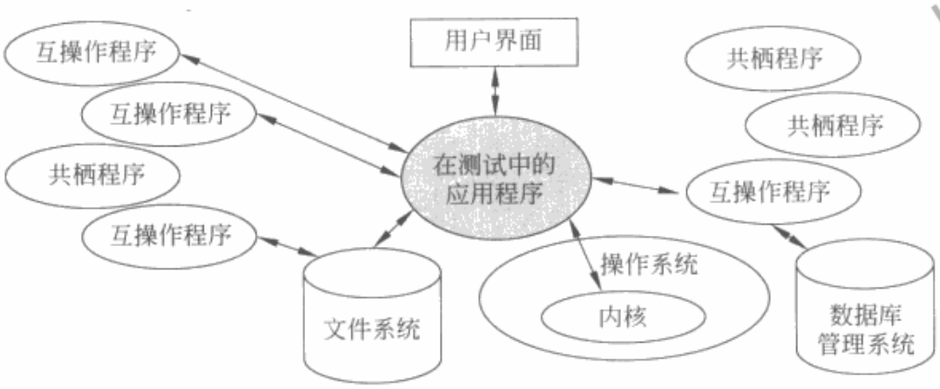


图 5-9　攻击依从性和 UI

介。因此根据上图，我们可以从其依从性和用户界面（UI）下手对其进行攻击。

依从性攻击包括：

- 我们可以阻断应用程序需要使用或加载的资料库的途径（例如，动态链接库）。
- 如果是在 Windows 操作系统上使用该应用程序，我们可以操作注册表，在该结构中编辑不同的关键码；如果是在 UNIX 和 Linux 操作系统中，也可以在相似的信息库中进行类似操作。
- 我们可以强制使用错误文件，尤其是当我们知道如何能够破坏这些文件时。
- 我们可以编辑数据和配置文件，或者将这些文件替换为那些用户无法通过有安全保障的界面而进行输入的值。（您可能会感到非常吃惊，居然在非常多的情况下，程序是处于前门即用户界面封闭，而后门大开即存储的配置、数据等易被攻击的状态。）
- 我们还可以强制在内存、CPU 空间，硬盘空间等可用资源很低的情况下进行操作。应用程序处理速度降低后，它可能会表现出一些它本来不该表现出来的状况。

用户界面攻击包括：

- 我们可以使输入数据溢出。在这点上，输入缓冲溢出是书中经常提到的老把戏，但是如果使用得当，至今仍不失为一种很好的测试方法。即使您不知道如何植入恶意的代码，那么只是在输入域中输入很长的代码就能使应用程序崩溃，那么这就是安全漏洞。
- 我们还可以尝试不同的程序切换或者是不同的配置选项，尤其是那些与使用特权和安全性相关的选项。
- 我们还可以试验不同的字符组、特殊字符、命令、脚本语言和数据库语言。众所周知的 SQL-injection 攻击就是这种试验的一个变形，它就是将一条合法的 SQL 数据库语句放入输入域中，结果便会导致该应用程序弹出一些本该受到保护的私有信息。

在图 5-10 中，您会看到在测试中的应用程序的设计和执行，它接受输入信息，执行计算，数据在数据存储器中转入转出（可能对文件系统或是数据库管理器也是如此，但是我不在这里一一阐述了）。应用程序最终产出输出数据。

pin51.com 最新最全源码资源下载 拼吾爱

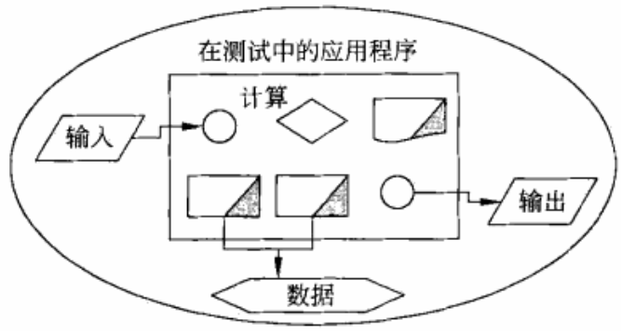


图 5-10 攻击设计和执行

因此，根据此图，我们可以在设计和执行时进行攻击。

设计攻击包括：

- 我们可以输入常规的账户和密码，包括一些经过良好设计的、但安全程序甚至都不能允许作为输入的账户和密码。
- 我们可以攻击可能存在的未加保护的 API。
- 我们可以试图依次接入每个端口，并攻击该端口。
- 我们可以设计循环，尤其对那些包含类似 TCL、SQL 的脚本语言的程序。
- 我们可以尝试使用特殊的工作流来完成一些任务。
- 我们也可以强制重置应用程序。

执行攻击包括：

- 我们可以对时间进行操作。我们可以意外地将其向前或（最好是）向后调整，或是使两个需要使用同一时间的系统时钟不同步。
- 我们可以试着复制需要高级使用权限的文档，尤其是脚本文件或是任何可执行文件，例如数据库命令。
- 我们可以强制触发所有（应该出现）的错误信息，以发现安全漏洞。
- 如果能找到的话，我们还可以寻找或进入临时文件，看看它们是否会保存有未加密的数据。这个工作包括交换空间和临时空间。

让我们来看一个例子。

在图 5-11 中，可以看到关于 Adobe 的 PDF 文档“写”功能的安全设置选项。我们可以设置密码级别的安全性，使之与各个 Adobe PDF reader 的版本都能够兼容，然后对文档所有、一部分或是少量的内容进行加密。我们可以允许或是不允许打印和复制文档内容。我们可以启用或是禁用例如文本转换为语音的协助软件。您可能需要花些时间研究一下这里的各个选项，想想如何攻击它们。

这时，您会看到我将打开、编辑和打印文档的功能都设置了密码保护。该软件接受了这些密码，并允许进行一些其他选项。但是遗憾的是，当您单击 OK 按钮时，会弹出如图 5-12 所示的信息。这是很明显而低级的安全漏洞，而不仅仅是我们通常提到的安全瑕疵，因为这个应用程序告诉了您它为什么存在这个瑕疵。接下来，您只需要上一下网，找到这类工具，就可以开始攻击 Adobe PDF 的文档了。

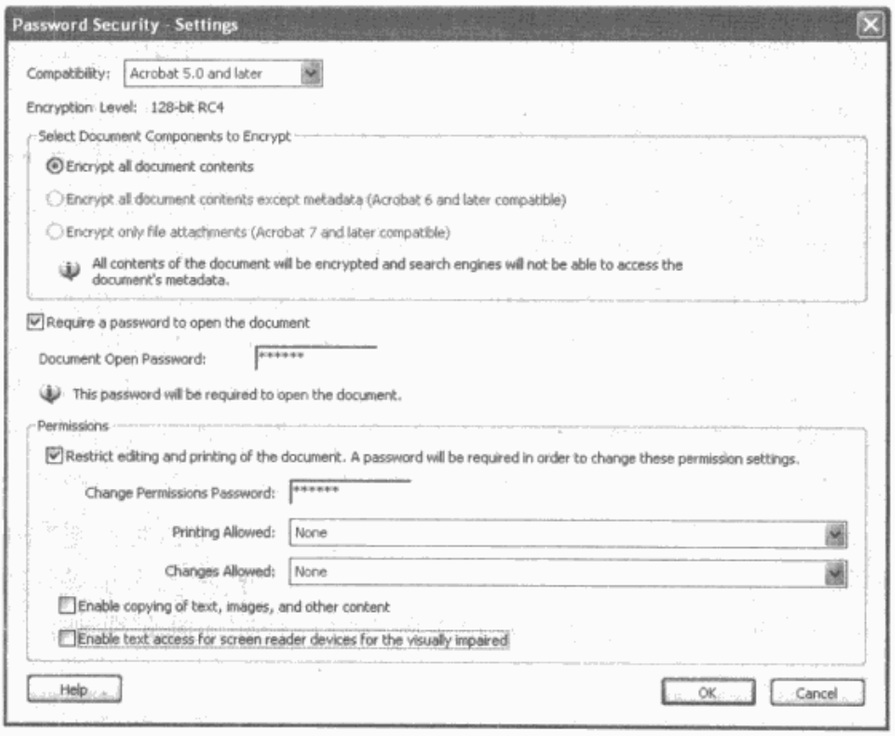


图 5-11　安全设置攻击举例

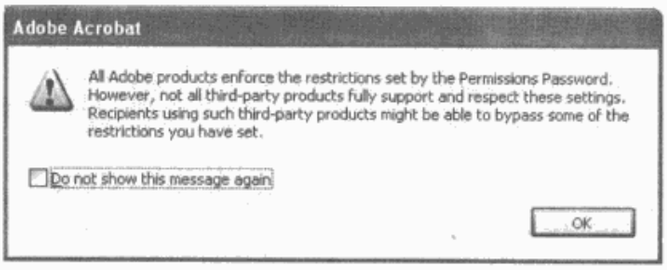


图 5-12　“如何发现安全漏洞”的举例

5.3.3　可靠性

可靠性对关键任务，安全为主和高使用率的系统是尤为重要的。我们可以利用可靠性测试降低可靠性问题带来的风险。层出不穷的系统漏洞证实，可靠性失效引发的常见的缺陷有内存溢出、硬盘碎片和耗尽、间歇性的底层结构问题以及低于可能的超时的值（lower-than-feasible timeout values）。

ISTQB 术语

可靠性增长模型：显示随着对系统或者组件进行连续测试，从而消除那些导致可靠性失效的缺陷，而使可靠性增长的模型。

可靠性：软件产品在一定条件下（规定的时间或操作次数等），执行其规定的功能的能力。

可靠性测试：判定软件产品可靠性的测试过程。

pin51.com 最新最全 拼吾爱

在可靠性测试中，我们监控软件的成熟度，将其与期望的、统计上验证的目标进行比较。正在测试中的系统有各项指标，例如平均失效时间（Mean Time Between Failures，MTBF），平均修复时间（Mean Time To Repair，MTTR），或是其他根据时间间隔或强度来统计失效数量的度量。在可靠性测试期间，当我们发现的缺陷由编程人员进行了修复，我们可以认为可靠性得到了改进。我们可以使用各种数学模型，也就是可靠性增长模型或是软件可靠性增长模型，监控在可靠性方面的增强。

有很多技术用来加速硬件可靠性测试，叫做高加速寿命测试（Highly Accelerated Life Test，HALT），软件可靠性测试通常还会涉及到延长持续时间的测试。该测试可以是一组工作流非常相似的命令反复运行。这种方法通常在工作流非常相似的时候起作用（例如，您可以在 ATM 上运用这种方法进行可靠性测试）；测试也可以由不同类型的测试组成，随机选取，但是前提是这些测试彼此之间的区别是可以被预知或限制的（例如，您可以在一个电子商务系统上运用该方法进行可靠性测试）；测试也可以是利用一些凭空想象出来的统计模型，也叫做随机测试（例如，一个电话交换机，由于被叫号码和呼叫时间等类似值上的变化非常大，您就可以运用这种方法对它进行可靠性测试）。测试数据也会根据不同的模型随机产生。

因为这是应用于高风险系统，如您所能想象的，一门精准的数学学科在这个方面应运而生。这包括了系统模式应用的研究，有时也被称为运行概况（operational profiles）。可靠性增长模型和其他一些存在于可靠性测试中的数学要素则应该根据经验数据进行微调，否则其结果就没有太大的意义。

ISTQB 术语

运行概况：由组件或系统执行一系列不同类别任务的代表值。这有可能是基于与组件系统交互时的用户行为以及它们发生的概率而产生的。任务是逻辑的而不是物理的，所以能跨多个机器执行或在非相邻的时间片段内执行。

运行验收测试：验收测试阶段的运行测试通常是由操作员或者管理员在一个模拟实际运行环境的系统中执行，关注软件运行方面的行为。例如：可恢复性、资源行为易安装性及是否符合技术标准。

运行环境：指在用户或客户现场所安装，也就是被测组件/系统将要使用的硬件和软件产品。软件可能包括操作系统、数据库管理系统和其他应用程序。

另外，由于标准工具和脚本技术的存在，可靠性测试通常是自动化的，我从未看到一个手动可靠性测试的例子，除非是自欺欺人。

根据可靠性的目标级别，我们可以将可靠性测试和度量当作退出准则。换句话说，我们会持续测试，直到我们达到了足够的可靠性级别。在某些情况下，服务水平协议和合同中会明确做到什么程度满足这个“足够”的定义。

特殊的可靠性测试类型包括健壮性和可恢复性。健壮性测试包括故意使系统在负面、有压力的条件下运行，看它做出何种反应。这其中可以包括耗尽资源。

pin51.com 华章图书 华章教育 华章分享

ISTQB 术语

可恢复性测试：判定软件产品可恢复性的测试过程。

可恢复性：软件产品出现失效（failure）后，其重建一定程度的运行能力，及其恢复受影响的数据的能力。

可恢复性测试主要是关心系统在其操作环境下从一些软、硬件失效中恢复功能的能力。失效备援测试（failover test）就是应用于存在冗余组件的系统的一种可恢复性测试，即确保一个组件失效后另外一个组件可以自动接手失效组件的工作。磁盘冗余阵列（RAID）就是很好的一个例子，而对备用数据中心的灾难恢复测试也是一个很好的例子。在失效备援测试中，我们强制执行一些操作让各种类型的失效发生，再检查系统恢复的能力，最理想的状况下，系统可以悄无声息地自己恢复。我所知道的最大的曾经运行过的失效备援测试是，基地组织（Al Qaeda）于 2001 年 9 月 11 日在美国的财政金融系统进行的。结果大部分系统恢复良好，但是在某些必要地点的后备员工的缺乏是当时较大的一个问题。因为就像在安全性方面一样，人的因素往往是容易被忽视的。

另外一种可恢复性测试是备份/恢复。它测试那些用以减少失效影响的流程和设备。例如，当数据丢失或被破坏时，有没有一种成文的、可重复的方法可以备份并恢复这部分数据？我听说一个首席信息官（Chief Information Officer，CIO）曾经因为有相应的规程却从不测试它们，而被炒了鱿鱼。

如果您需要测试备份/恢复功能，请确保您的测试程序中涵盖了所有重要的路径。您应该在动态测试的同时进行类似评审和审查的静态测试。理想的话，您应该在与产品实际应用相似的环境中运行这些测试，同时，对数据量敏感的系统，您还应该使用与实际应用相类似的数据量级进行测试。

在备份/恢复测试中，您可以测量以下变量：

- 执行不同类型（例如，全备份、增量型备份）的备份所花费的时间。
- 数据恢复的时间。
- 数据备份的保证级别（例如，恢复 24 小时以内的所有数据，恢复一个小时以内的指定交易数据）。

让我们来看一个例子，图 5-13 摘录自一份关于高容量互动语音应答（Interactive Voice Response，IVR）服务器的可靠性测试计划。

图中第一段定义了一些很重要的事项，即什么样的情况可以视为失效。关键是，任何数量的缺陷都可能触发失效，但除非这些失效会导致操作系统或是应用程序崩溃，或者是需要更换硬件，重启服务器等，该失效会得以计算，否则是不计算在内的。事实上，在该计划中的其他段落也讨论到了这点，足以说明在这一问题上的定义是非常明确的。

第二段定义了至少平均失效时间不小于一年的目标。

第三段描述了测试合格【这里被称为“验收”】的判定标准。这是对测试的一个验收准则。如果，在启动负载生成器之后，测试能在 114 天内无失效运行，就意味着测试完成，MTBF 目标达成。如果有一个失效发生，我们需要接着进行为期 164 天的测试。如

pin51.com 最新最全资源 拼吾爱

失效（对于可靠性而言）在需求规格说明书1.7版本中被定义为“……【失效】是IVR系统（操作系统【或】应用程序）崩溃【或需要】更换一些硬件【或导致】IVR重启。”

这些计划是为了展示【IVR】系统的MTBF等于或大于1年。

验收准则（仅计算计费失效）：

1）114天内0失效；或2）164天内出现1个失效；或3）214天内出现2个失效；或4）265天内出现3个失效；或5）315天内出现4到6个失效。

拒收准则：

1）17天内出现2个失效；或2）68天内出现3个失效；或3）118天内出现4个失效；或4）168天内出现5个失效；或5）218天内出现6个失效；或6）315天内出现7个失效

图5-13 可靠性测试举例

果两个失效发生，我们需要接着进行为期214天的测试——直到达到一个不合格的判定标准。您会发现随着失效数增加，我们需要进行测试的天数也随之增加。

第四段描述了测试不合格【这里被称为“拒收”】的判定标准。这是对测试的另外一个准则。如果启动负载生成器之后，测试在17天内出现两个相关、但彼此独立的失效，那么意味着测试结束，因为MTBF目标无法达成，我们将会拒收该系统。如果在为期68天的测试有3个失效发生，同样的，我们会拒收该系统，依此类推。

作为一个依靠统计学的结果，实际上是有可能会得出错误的结论的。通常会有两种错误的结论，就是需要拒收的时候您却接收了，需要接收的时候您却拒收了。在这个计划中，还有一些讨论是关于如何平衡这两种错误结论带来的风险的。这两种风险也分别被称作是客户风险和生产者风险。

5.3.4 效率测试

效率在很多情况下都是很重要的。时间关键的系统，包括大部分的安全关键系统、实时系统和任务关键的系统，都需要在规定时间内实现相应的功能。即便是类似于电子商务和销售终端系统这样的不那么关键性的系统也同样要求有较快的响应时间。

此外，对于一些系统，像是实时系统、消费性电子和嵌入式系统，资源的使用都是很重要的。您不可能总是在资源紧缺的时候增加硬盘或内存，例如NASA（National Aeronautics and Space Administration，美国国家航空和宇宙航天局）就在执行的一个火星任务中，发现过存储空间耗尽的类似情况。

效率失效包括响应时间过慢、数据吞吐量不足、在特定负载下可靠性失效，以及需要消耗过大的资源。效率缺陷往往是因为设计存在缺陷，所以在后期测试再去修正已经是很困难的了。因此，您可以也应该在各个测试级别进行效率测试，尤其需要在设计和开发阶段进行这些测试（可以通过评审和静态分析）。

ISTQB术语

效率测试：确定测试软件产品效率的测试过程。

效率：一定条件下根据资源的使用情况，软件产品能够提供适当性能的能力。

让我们从响应时间测试开始讨论关于效率测试这一话题。响应时间测试通常也被称为性能测试。在实际操作中，有各种不同的侧重类型，通常有：性能（当响应时间是问题关键时）、负载（当在不同负载级别下的响应时间是问题关键时）、压力（当在最大负载或高于最大负载时的破坏行为是问题的关键时）和可扩展性（当负载的限制是问题的关键时）。

经典的性能或响应时间测试一般会关注于，在特定的时间周期内，在各种合法条件下，组件或系统响应用户或系统输入的能力。有关这个问题，也可以从吞吐量的角度进行分析，这里吞吐量是指在指定时间内完成的功能数、记录、或交易完成。度量也会因为测试目的的不同而有所变化。

负载测试涉及不同级别的负载及其组合，这里的负载通常集中于研究可预见的并符合实际的负载。负载通常被设计为由一定数量的平行用户提出的交易需求。我们可以在执行测试的同时测量响应时间或数据吞吐量。有些人会把负载测试区分为多用户负载测试（考虑实际的用户数）和容积负载测试（Volume Load Test）（考虑大量用户），不过我通常不会这样区分。[2]

压力测试通过达到，并超过最大负荷和容量来进行极端的负载测试。该测试的目标是，确保在超出最大负荷和容量的情况下的响应时间、可靠性、同时保证功能衰减缓慢且同时是可预见的，最后以出现一些类似于“走开！我很忙！”这样的信息而告终，而不是应用程序或操作系统崩溃、锁死、数据毁损或是其他危害严重的失效模式。压力测试的一个变种是“翻转测试”，该测试的目的是为了找到完全饱和或失效发生的那个点。这个资源被耗尽点，也就是系统的薄弱环节。（理想情况下）通过变更设计或是（如果需要的话）添加更多的硬件可以提高处理压力的能力，有时甚至可以改进响应时间。压力测试的另外一个变种是“反弹测试”或“尖峰测试”，主要包括快速实现压力上升和下降，并循环进行。

最后，可扩展性（scalability）是压力测试的一个延续，主要是指找出系统的瓶颈，然后测试系统哪里可被增强，以解决瓶颈问题的能力。换言之，如果解决用户增长的方案是向服务器增加更多的 CPU，那么我们就可以通过可扩展性测试来验证该方案是能够满足需求。明确系统存在的瓶颈之后，压力测试也有助于产品建立负载监测管理的临界值。

资源利用率测试主要是评估，在正常情况和压力负载下，系统资源的使用率。资源包括内存容量和带宽、硬盘容量和带宽、CPU 容量和网络带宽。进行这类测试，尤其是对于嵌入式系统，需要将精力集中在那些核心的资源上，这些资源通常包括那些有限的或是最关键的资源。

与可靠性测试相类似，效率测试也会根据实际或预期的使用情况，经常涉及到运行概况这一概念。这些运行概况是实际用户如何与应用程序交互作用的模型。它们会根据一天的不同时间段、一周中的不同星期几、一月的具体哪一天和一年中具体的第几周甚至是哪一天的不同而发生变化。同可靠性测试一样，如果我们的运行概况是错误的，那

2　参见 Steve Splaine 的 The Web-Testing Handbook 一书。

pin51.com 最新最全 拼吾爱

么我们的效率测试的结果也是无意义的，而且很有可能还会误导我们。同样地，运行概况也是创建基于自动化工具的测试用例的关键输入。再次强调，手动的性能测试通常是有误导性的，对于那些支持大量用户和交易处理能力的系统来说就更是如此。

5.3.5 可维护性测试

可维护性是指能够更新、修改、再利用和测试系统的能力。该特性对大多数系统都是很重要的，因为大多数系统都会在其生命周期内被多次更新、修改和测试，而且通常部分甚至全部系统都会被使用于不同或者是全新工作情况下。

> **ISTQB 术语**
>
> **可维护性测试**：测量软件产品的可维护性的测试过程。
>
> **可维护性**：软件产品易于更改以便修正缺陷、满足新的需求、或者使以后的维护更简单的能力，或者该软件产品适应新的环境的能力。

可维护性的缺陷包括：难以理解的代码、环境的依从性、隐藏的信息和状态以及过分复杂。甚至包括，当软件还没有在该领域有可行的更新机制时，便被发布而引发的问题。举个例子来说，我们可以回想一下那些关于安全问题的补丁程序中在2000年年中给微软带来的影响。

可维护性测试肯定是包括静态分析和评审的。大多数可维护性缺陷在动态测试中是看不到的，但是在类似于代码分析工具、设计和代码检查、或者类似的方法中却很容易被找到。

这并不是说，您不需要测试各类更新、补丁、升级和迁移，这些测试肯定是应该进行的。而且，不仅要测试软件部分，而且还要测试涉及到这部分软件的过程和基础结构。例如，您用一个大小为100MB的补丁文件为系统打补丁，打补丁的过程没有任何差错，系统也工作良好。但是当您发现，您忘记考虑实际用户需要通过拨号连接下载这个补丁，而且有一半的补丁的文档需要一个计算机科学专业的博士进行手动安装时，事情就不像您想的那么好了。这也就是您在可维护性测试中需要验证的目标。

有时，项目或产品的度量会指出与可分析性、稳定性和可测试性相关的问题。例如，大量的回归测试失败就表明存在稳定性的问题；因缺陷导致的关闭时间过长（换句话说，也许某些程序员永远也找不到存在缺陷的问题所在，从而导致不得不长时间关闭。）可能表明存在值得分析的问题（当然这要视开发团队人员的水平而定）；太长的测试周期可能表明存在系统可测试性方面的问题（当然这还是要视测试人员和测试策略而定）。

5.3.6 可移植性测试

最后，可移植性是指应用程序可以在不同的环境中安装、使用，并在不同环境间迁移的能力。当然，前两点对所有系统都是很重要的。对于PC软件来讲，由于操作系统、共存和互操作的应用程序、硬件、可用带宽等的变更速度越来越快，能够被移植到新的环境中并适应新的环境，也成为尤为关键的软件特性。

pin51.com 拼五网 拼团 免费源码 分享大全

ISTQB 术语

可移植性测试：判定软件产品可移植性的测试过程。

可移植性：软件产品在不同硬件或软件环境之间迁移的方便性。

有很多类型的缺陷会导致可移植性的问题，不过，环境的依从性、资源的长期占用和不规范的操作系统交互当然是在这些缺陷中首当其冲的类型。例如，在安装时更改共享注册表中的关键值，或是在卸载时删除共享的文件都是典型的可移植性问题。

幸运的是，可移植性缺陷可以通过简单直接地运用一些现有的测试设计技术来修复。这些技术包括结对测试、分类树、等价类划分、决策表和基于状态的测试。然而，不幸的是，可移植性缺陷的测试通常需要进行大量的配置工作。

让我们来看几类不同的可移植性测试。

第一个是可安装性。在可安装性测试中，我们使用标准的安装文档、更新和补丁功能将软件安装到目标环境或环境群中。通常，在该测试中不仅仅需要执行一个测试用例，而是需要一整套的测试用例。我们需要检查的问题如下：

- 无法按照安装手册或是用户手册中的指引或是通过安装向导安装软件。听起来好像很简单，但是需要注意的是，这需要在不同的环境下进行测试，令您有足够的信心保证。该安装过程即便不是在所有的最终安装环境，但起码在大部分的环境下都是可以正常工作的，同时还需要考虑不同的安装选项，如：部分安装、典型安装、或全部安装。
- 如果不能正确地清除安装过程中的失效（例如，不能加载特定的 DLL 文件），将使系统处在一种未定义、崩溃、甚至全部损毁的状态下。像我们之前提到的，因为其存在太多的变化的可能性，使之成为一大挑战。
- 无法部分安装、无法中止安装或无法卸载。
- 安装过程或安装向导无法使我们免遭无效硬件、软件、操作系统、或配置的损害，或者根本就没有探测到这些可能的损害。这种情况很有可能与在安装过程中或之后产生的失效有关，进而导致系统处于未定义、崩溃、甚至全部损毁的状态。
- 完成安装耗费的时间令人无法忍受，甚至有可能根本就无法完成。
- 在成功安装或是安装失败后，无法降低版本或卸载。
- 参与安装人群普遍不知道如何正确安装，因此他们很迷惑、沮丧、而且会造成很多错误（进而导致系统处于未定义、崩溃、甚至全部损毁的状态）。很明显，这类问题通常会在安装的可用性测试中暴露出来。

顺便提一下，对于我刚刚提到的各类问题，除了安装，我们同样要考虑关于更新和补丁的问题。

这些测试不仅要求监控安装、更新、或打补丁的过程，而且要求之后再进行一些功能测试以检测是否存在被无意引入的问题。

接下来是共存测试，也被称为群居性测试或兼容性测试。在共存测试中，我们主要是检查一个以上的系统在同一环境下工作是否存在冲突。请注意，这与先前讨论的互操

pin51.com 最新最全 拼吾爱

作性的概念并不相同，因为这些系统之间不存在直接的交互的关系。之前我曾经把这个称之为“共栖”系统，这个用词可能存在一点点的误导，因为人类的共栖关系还是存在一定数量的直接互动的。在这类测试中，我们主要检查是否存在以下问题：

- 在同一环境下，应用程序之间在功能上存在负面影响。这种影响可以是直接的（使其崩溃），也可以是间接的（消耗所有的资源）。
- 应用程序一开始工作正常，但由于一些无法明确的依从性，在为其他程序打补丁和升级时，该应用程序遭到了损坏。

测试应用程序本身及其兼容性往往很容易被遗漏的。这些情况尤其在竖井管理结构的组织中，即当应用程序的开发是在不同开发小组分别进行时经常发生。尽管您可以在所有的东西都进入到数据中心时，再去在产品上进行兼容性测试，但是这并不是一个非常好的主意。

顺便提一句，在大纲中曾经提过，兼容性测试通常是在系统测试和用户验收测试已成功完成后进行。但是这并不是一个很好的建议，因为您可能会在项目结束时需要面对很多难以解决的问题。严格地来说，共存测试不应该晚于系统测试。

接下来是适应性。在适应性中，我们会检查应用程序是否能在所有预期的目标环境中正常工作。令人困惑的是，这个好像也是指兼容性测试。事实上，我更多的听说该测试被称为兼容性测试，而不是群居性测试或共存测试。您也许会想象到的，当存在很多选择时，仅指定适应性测试或兼容性测试的测试技术为结对测试、分类树和等价类划分就足够了；您可能还会想到，您需要在具体环境中安装应用程序，这样的话，应该可以同时进行适应性和安装测试，然后在该环境下对已安装和配置好的应用程序进行功能测试。通常，功能测试的少量样本即能暴露存在的任何问题，或者至少（通过暴露的问题）测算出该组织能够承受的，解决该任务潜在的巨大规模所需投入的（资源）。

我们可能还需要测试适应性中关于规程的因素。有可能数据需要从一个环境迁移到另外一个环境中，这种情况下，我们在测试软件的适应性的同时还要测试是否遵循了正确的规程。

最后，是可替换性。在可替换性中，我们要检查是否可以在同一个系统中，将软件的某些组件替换成其他的组件。这适用于使用商用现货（COTS）技术开发的组件的系统。例如，我们可能应该测试一个与数据库管理系统有接口的应用程序，从而保证我们还可以选择更多的数据库系统。

我们可以在系统测试中进行可替换性测试，但是首先，也可能是最好，是在功能集成测试中进行。如果在版本中存在一个以上的可替换的组件，常规来讲，我们都应该默认地进行这类测试。另外我们应该在评审我们的设计时注意可替换性的问题，因为可替换性正是一个清楚定义的界面接口所要求的质量特性。

5.4 认证考试模拟试题

作为每章的结尾，您可以试着做几道模拟试题强化学到的知识，帮助您对学习资料的理解，同时也可以为 ISTQB 高级测试分析师的认证考试进行准备。

假设您是一位测试分析师，正在测试一个家用可编程恒温（调节）器的项目，该产品可用于控制中央采暖、通风和空调（Heating，Ventilation，and Air Conditioning，HVAC）系统。除了普通的 HVAC 控制功能，该产品还能将数据下载至基于浏览器的应用程序中，进而再运行在 PC 上，进行进一步的数据分析。该程序还能将数据导出为 CSV、XLS、TXT 格式的文档。该应用程序还可以通过 AES 128 位、AES 256 位以及公钥加密技术为数据进行加密。

1. 以下哪项测试，测试了该应用程序的互操作性？
 A　加密数据使用公钥，选取适当的私钥是否能够将其破解
 B　评估典型用户可以从该应用程序中下载、分析数据的能力
 C　在操作期间定期将该恒温调控器的统计结果记录在一张纸上，然后检查应用程序输出的数据是否与之相符
 D　将数据导出到 Excel 格式的文档中，然后在 Office 的 Excel 软件中将其打开
2. 继续第一题的场景。以下哪项测试讲的是该应用程序的功能安全性？
 A　加密数据使用公钥，选取适当的私钥是否能够将其破解
 B　使用文本转换为语音的应用程序读取该应用程序屏幕上的数据
 C　在操作期间定期将该恒温调控器的统计结果记录在一张纸上，然后检查应用程序输出的数据是否与之相符
 D　评估典型用户可以从该应用程序中下载、分析数据的能力
3. 继续第一题的场景。以下哪项测试讲的是该应用程序的可达性？
 A　加密数据使用公钥，选取适当的私钥是否能够将其破解
 B　使用文本转换为语音的应用程序读取该应用程序屏幕上的数据
 C　评估典型用户可以从该应用程序中下载、分析数据的能力
 D　将数据导出到 Excel 格式的文档中，然后在 Office 的 Excel 软件中将其打开
4. 继续第一题的场景。以下哪项测试讲的是该应用程序的可用性？
 A　加密数据使用公钥，选取适当的私钥是否能够将其破解
 B　使用文本转换为语音的应用程序读取该应用程序屏幕上的数据
 C　评估典型用户可以从该应用程序中下载、分析数据的能力
 D　将数据导出到 Excel 格式的文档中，然后在 Office 的 Excel 软件中将其打开
5. 以下哪项测试是可靠性失效的最好例证？
 A　响应速度慢　　B　内存消耗过大
 C　突然终止程序　　D　数据加密失效

pin51.com 拼吾爱 最新最全源分享平台

第6章

评　审

> “只要我使用某个词，” Humpty Dumpty 以轻蔑的口气说，“那个词就能表示我想要表达的意思，一点偏差都不会有。”
>
> “问题是，” Alice 说，“您能否使词有这么多不同的意思。”
>
> “问题是，” Humpty Dumpty 答到，“哪个是最主要的意思，就这么简单。”
>
> Alice 十分迷惑，一句话也不说了。一分钟后，Humpty Dumpty 接着说：“词汇也有它们的脾气。有些词，特别是动词，是最傲慢的，您要小心使用；而形容词，您怎么使用都行。但奇妙的是，我掌握了所有词的用法。我说得就是这个意思”。
>
> “您能告诉我，” Alice 说，“那又意味着什么呢？”
>
> ——在 Humpty Dumpty 那令人不快的万有引力课开始前，Alice 和他进行了上面的谈话，看起来她的困惑并没有得到很好的解释。

高级大纲第 6 章介绍评审相关的内容。在基础级大纲中，我们了解到评审是一种静态测试，它是由人（而非工具）来分析项目或者项目的工作产品，如：需求规格说明。评审的基础级目标一般为，在工作产品成为下一步项目阶段活动的基础前，找出其中存在的缺陷。高级大纲会补充介绍其他的评审类型，以及成功高效评审的策略。第 6 章包括以下 5 小节：

1. 概述。
2. 评审的原则。
3. 评审的类型。
4. 评审的引入。
5. 成功评审的决定因素。

下面来学习每一节，并了解它们与测试分析是如何关联的。

6.1　概　　述

学习目标：

能回想起该部分内容即可。

在第 4 章的开头，我介绍过测试的分类（如图 6-1 所示，ATA 为高级测试分析师，ATTA 为高级技术测试分析师），也说明了静态测试与动态测试的区别。静态测试不需要执行测试对象；而动态测试必须执行测试对象才可进行。第 4 章和第 5 章，我们主要从动态测试的角度，探讨了测试技术和质量特性。

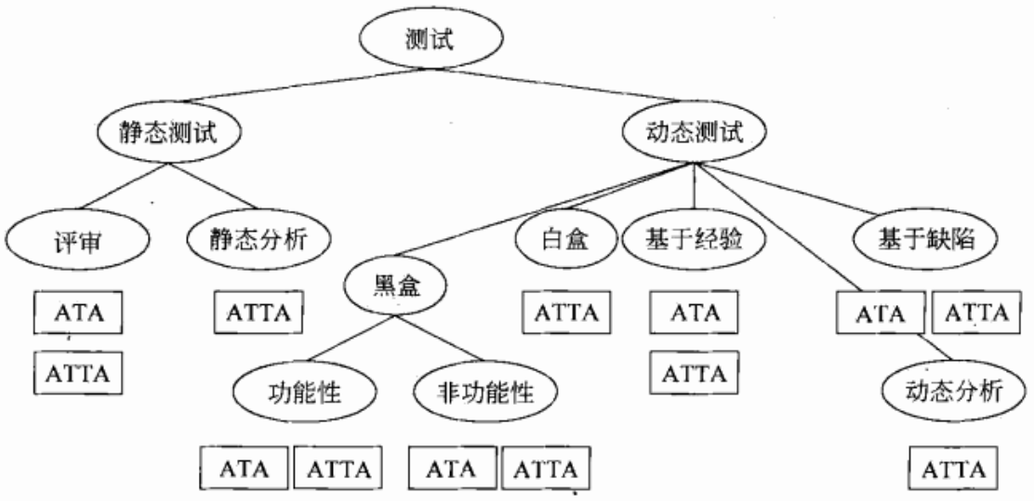


图 6-1 高级大纲中的测试技术

ISTQB 术语表

评审：对产品或项目状态进行的评估，以确定和计划的结果存在的误差，并提供改进建议。例如，管理评审、非正式评审、技术评审、审查和走查。

评审人：参与评审并辨别和描述被评审产品或项目中的异常的人员。在评审过程中，可以选择评审人员从不同角度进行评审，或请评审人员担当不同的角色。

本章侧重于静态测试，更确切地说，是静态测试的一个分支，即评审。（我们把静态分析留给那些高级技术测试分析师。）很大程度上，本章是基于基础级大纲中的评审内容展开的。

评审不只是人们坐在一起阅读文档。要想评审有成效，一个公司就必须在计划、参与和追踪等环节上有所投入，并做好这些环节。

优秀的测试人员本身就是优秀的评审员，因为他们具有好奇心，总能采用怀疑的态度提出各种问题（参考基础级大纲中的职业怀疑主义的态度）。这样的态度使他们在评审中非常有效。但是，最终他们还是需要保持清醒，用正面的方法为评审工作做出积极的贡献。

不仅是测试人员，每个参与评审的人都应该有这样的积极意识。评审是一个群体行为，所有参与者都应该致力于使得评审有序并积极发展。消极的、对抗的、或者是好战的参与者将会彻底破坏整个评审工作。

评审做坏很容易，但是做好却很难。这也是为什么许多公司直接放弃评审的原因。然而，如果做得好，评审将是与质量相关的各项工作中回报率最高的工作之一。

我们回顾一下在基础级大纲中介绍的 IEEE 1028 评审标准，如图 6-2 所示。标准的

pin51.com 拼吾爱 最新最全源分享

第一节是概述，包括制定评审标准的目的、覆盖的范围、符合评审标准的大纲、评审标准的组织和系统，以及如何在公司实施这些评审标准。

1. 概述
 - 目的，范围，合规性，组织，应用
2. 参考资料
3. 定义
4. 管理评审
 - 职责，输入/输出，入口/出口准则，规程
5. 技术评审
 - 职责，输入/输出，入口/出口准则，规程
6. 审查
 - 职责，输入/输出，入口/出口准则，规程，数据搜集，过程改进
7. 走查
 - 职责，输入/输出，入口/出口准则，规程，数据搜集，过程改进
8. 审计
 - 职责，输入/输出，入口/出口准则，规程

图 6-2 IEEE 1028 软件评审标准

第二小节介绍了参考资料，顾名思义，是指参考其他的文档、标准等。第三小节“定义”，是定义标准中使用到的术语。

第四小节介绍管理评审。您也许会注意到，基础级大纲中没有这方面的内容。在本小节中的标准给出了：在管理评审中，谁应该承担什么职责、管理评审的输入和输出、入口/出口准则以及管理评审应该遵循的规程。

第五小节介绍技术评审。在本小节中，标准给出了：在技术评审中，谁有什么样的职责、技术评审的输入和输出、入口/出口准则，以及技术评审应该遵循的规程。

第六小节介绍审查。和前两节一样，在本小节中，标准给出了介绍：在审查中，谁有什么样的职责、审查的输入和输出、入口/出口准则以及审查应该遵循的规程。然而，由于审查比技术评审更正式，标准中还讨论了有关从审查过程搜集数据，并在审查过程中应用相关改进这方面的内容。

第七小节介绍走查。和前几节一样，在本小节中，标准给出了：在走查中谁有什么样的职责、走查的输入和输出、入口/出口准则、走查应该遵循的规程。因为走查和审查同样正式，标准中也讨论了有关从审查过程搜集数据，并在审查过程中应用相关改进这方面的内容。

第八小节介绍审计。和前几小节一样，在本小节中，标准给出了：在审计中，谁有什么样的职责、审计的输入和输出、入口/出口准则以及审计应该遵循的规程。

6.2 评审的原则

学习目标：

能回想起该部分内容即可。

我们先来看一下基础级大纲中讲到的原则。首先，如前所述，评审是一种静态测试。在评审过程中，评审对象并不被执行。作为测试活动之一，评审的目的也有多种。基本目的是发现缺陷；另一个目的，在所有测试中都比较典型的，是建立信心，即我们能够继续推进被评审的对象，减少使用它的风险，为管理层提供信息；评审本身还有一个特殊的目的，即保证对文档及文档对项目的内在影响有一致的理解，并且对文档中的陈述形成统一意见。

评审通常先于动态测试，为动态测试做补充。缺陷在系统里存在的时间越长，产生的代价会越大，所以评审应该及早进行。然而，评审并不能发现所有的缺陷，动态测试仍然是必须的。

电影导演 Woody Allen 曾说过，“80%的成功在于自我展示”。这个说法在电影行业也许是对的，但他本人应该不会是一个有帮助的评审参与者。评审要求有充分的准备，否则将毫无意义。

实际上，如果在参与评审之前认真读了文档，您就不会轻易地问那些本可以自己回答的愚蠢问题。或许您认为这样说过于严厉，尤其是依据管理学中“不存在愚蠢的问题”这样的陈词滥调。但是，抱歉地讲，愚蠢的问题实在是很多。任何由于没有准备而产生的问题，如果导致满屋子的人，不得不看着别人花时间，给那个没做准备的参与者解释那些进会议室之前就该知道的事情，这样的问题足以称为愚蠢的问题。在我看来，不做任何准备就出现在评审会议上是一种无礼的行为，是对其他与会者的不尊重行为。

规范的评审能够很大程度地提高工作效率，每个公司都应该对其所有重要文档进行评审，包括测试文档、测试计划、测试用例、质量风险分析、缺陷报告还有测试状态报告，只要您想得到的文档都该评审。我的经验是，重要事情除非已经被两双眼睛检查过了才叫完成。您不必去评审无关紧要的文档，但问题是：您为什么要写那些无关紧要的文档呢？

评审之后会发生什么？有 3 种可能的结果。理想状况是文档质量很高，没有任何问题，或只要做很小的改动。另一种是，文档需要修改但不用再次评审。最昂贵的结果，从工作量和时间安排上说，是文档需要大范围的修改，并且还要重新评审。假设第三种情况出现了，记住，尽管它是三者中花费最大的，如果您简简单单跳过严重问题，紧接着就进行组件、集成、系统或者（更糟糕地）验收测试，那么您付出的代价将会更大。

非正式评审没有固定的规则、角色或责任，您怎么做都行。当然，记住 Capers Jones 曾总结的，非正式评审一般只能找出大约 20%的缺陷，而非常正式的评审，例如审查，能发现达 85%的缺陷。对于重要的东西，您或许该进行正式评审，除非您和您的团队已经聪明到不会犯任何错误了。

正式评审中，会包括如下重要的角色和职责：

- 经理：分配资源，安排评审时间等。但是，对于某些类型的评审他们是不允许参与的。

ISTQB 术语表

非正式评审：一种不基于正式（文档化）过程的评审。

审查负责人（或主持人）：负责审查或其他评审过程的负责人或主要人员。

pin51.com 最新最全资源分享下载 拼吾爱

- 主持人或负责人：评审会议的主席。
- 作者：编写评审对象的人。合格的评审会议不会使作者感到难受或出丑。
- 评审员：对评审对象进行评审，发现缺陷的人。评审员可以有特定的角色，基于他们的专长或他们要针对的缺陷的类型。
- 秘书或记录员：记录评审中发现的缺陷和改进意见等。

对于有些评审，同一个人可以担任不同的角色。比如，作者、主持人和秘书可以是同一个人。拿我来说吧，作为测试经理，当测试团队评审我的测试计划时，我一般同时担任经理、作者、负责人和秘书4个角色。

有些评审还可能涉及到其他角色，比如决策人或者项目利益相关者（stakeholder），特别是当评审的主要或者次要目标是为了达成一致意见或发布信息。有时，参与评审的利益相关者可以是客户或者用户代表。近来，我们对公司新网站的模拟网站进行了评审。市场团队、外包的网站开发团队和公司管理层人员一起参与了评审。网站开发团队是我们雇用的，所以，对于他们来讲，我们就是客户，从某种角度看，也是用户。

某些特别正式的评审也会用到阅读人员（reader），负责阅读和解释被评审对象。

一个公司可以使用多个评审类型。对于一些文档，节省时间比追求完美更重要。例如，对于某测试人员的缺陷报告，我们的测试团队使用“两双眼睛”的原则，即，必须由另一个测试人员审阅通过才可归档；但是，对于可视化较好的文档，比如测试计划，我们则在整个测试团队使用走查的方式；而对于重要的文档，我们不止使用单个评审方式。比如，写作这本书时，我们和RBCS的朋友及合作伙伴一起进行了非正式的评审；然后，当整本书出来时，又进行了更广泛和正式的评审。

6.3 评审的类型

学习目标：

（K2）比较不同类型的评审，说明它们各自的优势、劣势和适用领域。

基础级大纲里讨论了4种评审类型：

- 最低级别的评审（通常能有效排除部分缺陷），我们称为非正式评审。非正式评审可以简单到两个人，如作者和他的同事，电话讨论某个设计文档。
- 技术评审较为正式，但不是最正式的。技术评审有时也称为同行评审，这个类型的名称本身已经说明了一般谁会参与评审。
- 走查这种评审方式中，作者就是主持人，评审过程就是对评审对象逐步检查。也就是说，作者主导评审的进行，评审人员一段接一段地走查被评审内容。
- 审查是最正式的评审。审查中人员角色又明确定义。经理不需要参与审查。作者既不做主持人也不担当秘书角色。通常还需要阅读人员。

可以看出，越是正式的评审，评审的速度——每小时评审的页数——越慢。

应当记住的是，IEEE 1028标准和基础级大纲中讨论的是理想情况。现实当中您会发现，公司往往混合使用他们青睐的评审类型，放弃他们不喜欢的方式。同样，您也会

常常听到，许多公司的走查方式并不符合 IEEE 1028 规范。

基础级大纲没有涉及到 IEEE 1028 规范中的管理评审和审计。接下来我们就一起学习这方面内容。首先介绍管理评审。管理评审的基本目的是监控过程、评估状态，对某些项目、系统、正在进行的活动或过程做决策。（当然，也有些公司会由于各种政治原因组织管理评审，在此，我们不考虑这样的情况。）

通常，评审对象所属部门的经理执行管理评审。各种利益相关者和决策人也会协助他们。每个参与者的参与程度可能不同。有时，公司可雇请外来顾问做评审。比如，RBCS 很大部分的业务来自为其他公司做各类测试过程和质量过程的评估。这些模式混合了管理评审和后面要探讨的审计。一般由测试经理推动测试评估，这种情况下，测试评估更接近于管理评审，但如果是由外部利益相关者推动，则更像审计。

管理评审的一部分工作是从计划、报表、项目风险等方面评估项目的状态；另一部分工作是查看各方面的规程，并监控规程的充分性。评审参与者必须做好准备工作，特别是那些要报告状态信息的人。我们曾为某机构做测试评估，他们的人员对测试团队的进度知之甚少，于是我们给出的最大建议是：“立即实行跟踪机制。”

管理评审的输出通常包括行动、建议、待处理问题等。管理评审做出的决策应该文档化，并对行动和建议的执行进行例行检查。不幸的是，后续工作常常做得不理想。

下面我们来看审计。审计是非常正式的，有时带有对抗性的。在审计中，审计人员检查人们的行为是否与合同、标准、行业最佳实践等相一致。审计可能激起自我保护。高级大纲中提到，在发现缺陷方面，审计的效率最低，但这实际上取决于审计活动和被审计的组织。我们在为客户做测试审计时，就非常善于发现缺陷，不论是项目缺陷还是过程缺陷。

审计的基本特征是独立评估。如同管理评审，我们可以衡量某个过程、项目、当前活动或者系统。审计另一个关键特征是检查是否符合标准。审计可以由一个审计团队或者某个审计员完成。审计员通过谈话、视察、检查文档、分析数据等方式搜集证据。我发现谈话有时非常有趣，特别是某些谈话对象已成为高级投资人的时候，他们会试图欺骗、误导、说服或拖延审计人员。

类似于管理评审，审计的输出包括行动、建议、待解决问题等，另外还包括合规性（或者非合规性）评估的结果。合规性评估依据的是多维标准或检查列表，所以即使您有 99 项符合合规性，而有 1 项不符合，那么您还是不能通过审计。如果结果显示非一致，那么针对失败项的纠正行动要有代表性。审计的决策也应该文档化；行动、建议以及纠正行动的执行，连同合规性的定期再评估都应该例行检查。针对规范行业或者有明确法律条款约束的审计，其结果的后续工作通常很好，但是不规范行业的审计后续工作一般不理想。

除了 IEEE 1028 标准中划分的评审类型外，我们还可以从工作产品或被评审的活动角度对评审进行分类。合同评审，很自然地，和项目的某些转折点或重大事件有关，比如付款、合同续约。对于安全性要求很高或者安全关键系统，合同评审也可作为管理评审。合同评审的参与者可以是经理、供应商、客户和技术人员。如果项目进展顺利，合同评审只是例行检查。但是当项目遇到问题时，特别是如果涉及到多个供应商，大量时

pin51.com 最新最全 拼吾爱 下载

间和精力要花费在查出问题的责任人。

基础级大纲中讨论了需求评审。需求评审可以是非正式评审，也可以是走查、技术评审或审查。需求评审的对象范围应该是无限制的，不论什么需求都可以。对于安全关键系统，评审应该考虑到安全性需求；对于高可用性的系统，评审则要考虑到可用性和可靠性需求。不论什么系统，需求和需求评审既要关注功能性也要关注非功能性。需求评审可以考虑引入需求评审验收准则和测试条件。

基础级大纲中还讨论了设计评审。设计评审也可以是非正式评审、技术评审或审查。设计评审涉及到技术人员、客户和利益相关者。评审越是深入到细节，参与者越应该更偏技术。在某些公司，设计评审有初步设计和关键设计之分。初步设计评审采用同行评审，技术同行提出初步方案以解决系统或测试设计中的技术难题。关键设计评审则覆盖所有已提交的设计方案，有时还包括其中的测试用例和测试规程。

操作就绪评审、验收评审或资质评审是技术和管理评审的组合体，是最后的安全保障。通过评审所有的数据，最终决定是否开始生产。但是有这样的情况，项目处于非常紧张和疲劳的状态，到了验收评审时，不经审查就批准进行生产，尽管有很多理由表明“得再等等，现在还不是时候”。有一个项目出现了这种情况，最终导致生产中很长时间系统性能极差。

基础级大纲中介绍了正式评审由6个阶段组成：

1. 计划：包括在所有工作产品的评审，项目内的所有评审。
2. 预备会：包括在所有工作产品的评审，项目内的所有评审。
3. 个人准备：为每个工作产品做准备，读文档并记录问题。
4. 评审会议：为每个工作产品开评审会议。
5. 返工：依据评审结果的要求对被评审对象进行修改。
6. 跟踪结果：跟踪单个工作产品的修改，如果需要的话，对整个项目的评审后继工作进行跟踪。

好的规程固然重要，但是合适的参与人员也很关键。参与者必须了解被评审的工作产品。如果没有合适的评审人员参加评审，就算严格按照规程，评审也不会成功。

Capers Jones 在研究了几百家客户的数千个项目之后，发现关于评审、应用对象和评审有效性的一组有趣数字。Jones 注意到，非正式评审有效性最差，比较正式的评审有效性中等，高度正式的审查是最有效的。当然，不论正式程度如何，要想评审有效，就必须认真对待评审，并在规程上得到公司或部门的支持。[1]

从表 6-1 可以看出，评审的正式级别和被评审的对象，对发现和排除缺陷的比率有很大的影响。如果您把评审看做是一系列的过滤器，下面的例子说明了评审的有效性。

首先，假设有 1000 个缺陷。采用非正式的评审做法，但是评审了所有条目，那么评审后遗留缺陷为 166 个；如果采用最正式的评审做法（也评审了所有条目），那么评审后遗留缺陷只有 3 个。

1 表 6-1 和其中的数据来源于 Capers Jones 的 Software Assessments, Benchmarks, and Best Practices 一书。

表 6-1 评审类型及其有效性

	最 低	平 均 值	最 高
需求评审	20%	30%	50%
概要设计评审	30%	40%	60%
功能性设计评审	30%	45%	65%
详细设计评审	35%	55%	75%
代码评审	35%	60%	85%

6.4 评审的引入

学习目标:

能回想起该部分内容即可。

以下步骤将有助于评审的成功实施：

- 获取管理层的支持：从预算来看，评审不如测试自动化花费多，但是评审需要时间成本，特别是时间较紧时，需要得到管理层的支持。
- 得到经理们的认可：和经理坦诚交流评审的效益是必要的，包括成本、利润和潜在的问题，切忌夸大其词。
- 提供基本架构：将要使用的各种评审方式文档化，包括模板和表格。建立基本工具，比如评审信息数据库，特别是对于地理上分散的评审，要确保辅助工具到位。
- 培训：对参与者进行评审技术和规程的相关知识的培训。
- 获得参与者的支持：保证评审员和评审对象的作者支持评审活动。
- 试验性的评审：设置一些错误，进行试验性的评审，并从中获得经验教训。
- 展示评审的好处：当您有了一个成功的业务案例之后，向管理人员展示您的成果。
- 推广评审所有文档（至少对最重要的文档）：需求、合同、项目计划、测试计划、质量风险分析以及可视化高的文档都可以作为评审对象。但是我发现，仅仅是对缺陷报告做非正式的评审也能获得很多收获。

您不需要在每一个组织中实施上面的每一个步骤，也不需要逐个按顺序执行这些步骤并做到非常完美，但是当您想从中省略某个步骤时则必须仔细考虑是否可以省略它。

您的组织将在评审中投入时间和金钱。经理们期待这份投资有所回报。度量，例如通过测算减少和避免修复缺陷或者失效的成本，可被用来展示引进评审工作是否能够带来收益。试想，在系统测试中修复（一个）缺陷的代价是多少？在系统发布之后又将如何？一张简单的电子表格就可以显示出评审的好处并评估出在实施评审后的收益。

也不要忘记计算由于节省时间而带来的回报。对于经理们来说金钱不一定永远是最大的关注点。事实上，让产品尽快地上市通常是更大的焦点。在项目早期的评审中发现问题能够降低在项目后期发现问题而导致项目延期的可能性。例如：在评审中发现的缺

pin51.com 最新最全 拼吾爱

陷需要5小时的时间去解决它，而在系统测试中发现缺陷需要用25小时去解决它。

如果已经建立了度量元，持续地监测这些度量会很重要。让评审过程形式化是非常容易的，评审的价值自然会不断下滑。如果您发现评审的作用越来越低时，必须自我反省找出原因。事实上，评审的好处应该是持续不断地上升、增加的。您应该使用基于度量元的量化数据去改进评审过程。保证您和您的经理们能够将评审和评审过程的改进当成一项长期投资。

6.5 评审的成功因素

学习目标:

（K3）使用评审列表，从测试人员的角度对代码和架构进行检查。

（K3）使用评审列表，从测试人员的角度对需求和用例进行检查。

很多因素影响着评审的成功与否。高级大纲中把这些因素分为3类。首先从技术因素来分析。

确保评审正确地按照定义好的过程进行。这对于正式的评审（如审查）可能会更加困难。当然，这并不意味着您不可以定制评审过程，但是在定制之前最好能先掌握好基本步骤。

我提及过有关评审效益的问题。为了评估评审的效益，您必须记录评审所耗费的成本（特别是工作量）和所获得的益处。对于评审而言，所获得的益处不是立即可见的，而是随着时间的推移才逐渐地显示出来。这对于所有测试活动都是一样的。但是评审表现得尤为明显。

无需等到文档全部完成才开始评审。当在处理一些严重问题时，可以评审初期的草稿或部分文档，这有助于发现缺陷并防止缺陷被引入到整个文档中。

也就是说，要对评审设定相应的入口准则。否则，您可能会浪费人们的时间去评审一些因为不够成熟而不足以接受评审的材料。您也有可能因为让他们评审一些还在不断改变的资料，使人们感到沮丧，且浪费了时间。人们往往会由于在寻找（处理这些不断变更的内容的活动中）停止浪费时间的方法，而浪费更多的时间感到备受挫折。这也意味着评审过程逐渐失去意义。因此需要建立入口准则，其中需要包括一个简单的原则，即所有参与人员都必须有所准备。

检查表对于评审很有帮助。人们往往关注重要的领域而忽略了检查表。可以从业内知名的专家们那获得检查表，但是必须确保检查表能适用于特定的组织。检查表必须能够发现一些常见的缺陷。不同的文档使用不同的检查表，如需求、用例和设计。最后，为不同的评审过程创建不同的检查表。

由于评审的正式程度有所不同，所以需要考虑使用多种类型的评审。根据目标决定采用的评审类型：是否仅仅是在发给客户之前进行一个快速的文档清理？还是需要改进一些技术设计方面的决定？去引导或培训利益相关者？为管理层提供一些通用信息？

我提起关于“两双眼睛的原则”，并且我努力尝试遵循这个原则。有时，项目的时间

压力会破坏这个原则。不管怎样，需要评审，或者更理想的话，要审查那些至关重要的文档。如果一份文档包含一项重要的决定，如签署合同，那么一定要在决定之前审查建议书、合同或者更高级别的需求规格说明。如果正在计划一项重大的支出，则需进行管理评审并获得批准。

对于大型的文档，可抽样调查某一特定的文件子集来评估整个文档中的缺陷数。这将有助于决定是否需要再评审。牢记这种抽样调查并不适用于文档清理或编辑。

对于那些令人分心的事物需要非常当心。我们很容易发现很多关于格式、拼写和语法错误。但我们需要把注意力集中在一些重要的基于内容的缺陷上而不是拘泥于形式。

最后，如前所述，要持续不断地改进评审过程。

现在我们谈一谈有关组织方面的因素：

特别是在最后期限的压力下，也要确保经理有计划地进行评审，并留有足够的时间和预算。在软件工程领域中比较盛行一种错误的想法：你在测试过程早期跳过高效率的缺陷移除活动，工作进度会有所加快。这其实是一笔虚假的经济账。

对于度量元（的选取）要特别小心。一些评审会发现很多的缺陷，而另一些评审则不会。业界存在一些数学模型去预测缺陷数，这点已经超出了本书的范围。小心，不要去使用简单的模型。最重要的是，永远不要把评审缺陷度量用来进行个人绩效评估。这将导致整个过程以失败告终。

确保有足够的时间去修改缺陷。典型的错误的测试实践是假设一项测试活动不会发现缺陷。

确保合适的评审员参与整个过程。Motorola 在 2000 年年中的一份研究显示，合适的参与者对于评审成功与否至关重要。[2]这当然包括技术和主题（或领域）的专家。确保评审团包括所有核心小组的代表，这是关于评审团队是否具有良好平衡性的问题。同时，这还涉及到对于评审过程的理解。对于正式的评审这一步通常通过培训来实现。他们发现的第二大关键影响因素是需要有合适数量的参与者，所以我们需要仔细考虑谁来参与，多少人参与。

如果您所在的组织是中型到大型的且正在使用评审，那么可以考虑创办一个论坛，让大家分享各自的经验和想法。

让那些在评审中不做任何贡献的人参与评审是毫无意义的。因此，要保证参加者能对评审做出相应的贡献。做到这些一方面需要有恰当的准备工作，另外就是让话比较少的人参加会议，不要以为一个人没有强势的性格就没有好的想法。

再者，将最强大的正式评审技术用于最为重要的文档。记住 Jones 评审有效性中的数字，在每个文档中您可以接受多大比例的遗留缺陷？

确保评审过程改进中有过程可遵循。评审过程的改进必须有相关的度量数据的支持。确保评审过程改进的过程能够识别出取得的成果。

最后，谈一谈关于人的因素。

2 Jeff Holmes, "Identifying Code-Inspection Improvements Using Statistical Black Belt Techniques", Software Quality Professional，2003 年 12 月

pin51.com 最新拼吾爱

作为经理，应当告诉所有的利益相关者和评审员对缺陷要有心理预期。确保他们对发现缺陷不会有不必要的惊讶；确保他们允许返工和重新评审所需要的时间。当今的人们总是倾向给自己安排过多工作，在这种情况下，面对文档中一系列待解决的问题是一个痛苦的经历，因为这意味着加班。

评审主持人不是西班牙宗教法庭的大法官 Torquemada。拷问、镣铐和冲水板不是评审工具或者技术。对于作者来说评审应该是一种正面的经历，在这里可以从同行那学会如何把工作做得更好。我依然记得在我早年的职业生涯中，两至三位导师在评审活动中帮助我进步地很快。也就是说，如果作者有过负面的经历，一定要小心不要强迫作者同意评审。最好由管理层来处理这些事。

当我们发现缺陷并考虑到如何在评审中高效地定位并移除缺陷时，我们应该感到开心而不是沮丧。让人们认识到他们可以发现了一个节省时间和金钱的大好机会，而不是在发现缺陷时妄加指责或谴责。

关注房间里的谈话。我们希望有一场建设性、有益性和客观性的讨论。鼓励人们对文档进行深层次的思考，包括如何将评审中的文档适用于项目的更大范围。[3]

6.5.1 Wiegers 的评审检查表

Karl Wiegers 提供了两个有用的检查表，一个是对需求评审，一个是对用例评审。我把两者都列入在此。[4]当您阅读需求规格说明或用例时，可以牢记这些问题并相对应地检查文档。

让我们首先来研究一下需求检查表。在文档组织和完整性方面，检查表列出了如下问题：

- 内部的交叉式引用是否正确？如果我们在这个文档中引用另一个文档或部分，这个引用是否有效？这是很容易出现的错误，并且一旦出现将会令人非常迷惑。
- 细节的详细程度是否一致并准确？业务分析人员和市场人员很容易在他们的文档中把一些本来很简洁的描述转化成他们需要的但是非常绕口的描述。
- 需求是否为设计提供了足够的数据？当非功能性需求被省略时这是一个非常大的问题。
- 是否包括了需求的优先级？如果没有，将很难进行权衡和决定执行顺序。
- 是否定义了外部接口？
- 是否存在一些缺失的信息？如果是这样，是否明确地标记了 TBD，即“待定义”？
- 是否描述了期望的行为？很多次，人们常说：“噢，那是很明显的”。真的是这样吗？

在正确性方面，检查表列出了如下问题：

- 是否需求冲突或者重复？评估这一步意味着您必须多次阅读文档。
- 是否每个需求表达清楚、简洁，不含糊？

3 对评审比较正式的讨论，参阅 Tom Gilb 和 Dorothy Graham 的 Software Inspection 一书。

4 Karl Wiegers，Software Requirements。

- 是否每个需求可以验证或实现？是否可以设计测试用例去证明需求已经被实现？事实上，完善的、成熟的需求规格说明书就是应该实现这个目的的。
- 是否每个需求都在项目范围之内？超过项目范围是很常见的事。如果真的越界了，测试团队会受到很大影响，因为他们要做比预期更多的测试，并对计划内的测试也没有办法投入足够多的时间来完成。
- 是否每个需求都没有内容和语法方面的错误？当您在处理错误的语法时，往往很难发现其他的更加重要的错误。
- 所有的需求都能够在约束范围内实现吗？对于测试人员而言，这个问题很难回答，但这也依赖于您的经验。
- 错误信息是否唯一且有意义？
- 在质量特性方面，检查表列出了如下问题：
- 是否所有的性能指标进行了恰当的说明？
- 是否所有的安全因素进行了恰当的说明？
- 是否其他相关的质量特性目标都通过可接受的方式，已明确地文档化，并进行了量化。

事实上，我会把整个 ISO 9126 中与质量相关的特性列入到检查表中去查看是否有任何的遗漏，因为忘记非功能性的需求是再平常不过的事了。

在追溯性方面，检查表列出了如下问题：

- 是否唯一地并正确地标识了每个需求？如果不能标识需求，您将无法追溯测试至原始的需求。同时，也要确保需求（被描述、分解）的粒度要适中，以保证测试到需求的可追溯性。换言之，如果需求包含的内容太多，您则可能需要设计 2 个、10 个、甚至 20 个用例去测试这个庞大而复杂的需求。试想当其中的 1 个测试失败了，您该怎么办呢？
- 如果您在评审一份详尽的需求，是否每个软件的功能性需求都可追溯到更高级别的需求？

最后，检查表列出一些特殊的问题来让读者思考：

- 是否我们处在需求的王国里，而不是设计？换言之，是否所有的需求确实是需求而不是设计或者实施解决方案？
- 是否定义了时间紧急的功能，并且它们的时间标准已经定义？
- 国际化的问题是否得到了充分地关注？

通过参照检查表，您能发现需求中的许多缺陷。当然，在评审需求规格说明时，如果发现一些奇怪的信息，您不应该只是因为想不出如何将之与 Wiegers 的检查表相对应就简单地忽略它。检查表只是思维的参照对象，而不能代替思考本身。

现在我们来看看用例检查表。关于用例评审，列出了如下一些问题：

- 用例是否是一个独立的、分离的任务、还是我们需要结合两个任务？
- 用例的目标是否明确？我们在这里尝试做什么？
- 哪一步行动从用例中得到好处，是否已经清晰？好处是什么？

pin51.com 最新最全源码分享 拼吾爱

- 用例是否书写得像摘要而不是一个特定的场景？换句话说，它应该是一个逻辑的测试用例，而不应是具体的用例。
- 用例是否已经做到不包含设计和实施的细节？
- 预期的异常行为是否记录在文档内？如果没有，很可能发生的事情是未定义的，在处理上也就会没有根据。如果有记录，至少在出现如程序崩溃这样的异常时，有可参考的处理方法。
- 用例中的每一步行动与任务相关联？或者，我们是否需要在单个的用例中包含了多项任务？
- 用例中每一个已定义的工作流是否是可行的？是否确实能够实现？在这里也需要考虑例外情况。
- 用例中每一个已定义的工作流是否是可验证的？您能测试这个工作流吗？

通过参照检查表，可以发现用例中的很多缺陷。跟以前一样，当您评审用例时发现一些异常，就抓住它并检查是否与 Wiegers 检查表相关。

6.5.2 Deutsch 的评审检查表

评审应该运用到设计中，而不仅仅针对需求和用例。我们先看一个检查表范例，这个例子用于评审分布式应用。1990 年，Sun Microsystems 公司的员工 L.Peter Deutsch 和他的同事从工作中总结出这一检查表。也许人们还记着 Sun 早期的口号：“网络就是电脑”。他们作为分布式开发和设计的开拓者，Deutsch 和他的同事发现人们在开发和设计分布式应用程序时一直重复同样的错误。

下面是分布式应用程序设计中最主要的 8 大谬论，也即导致人们犯错误的一些并不成立的假设：

1. 网络是可靠的，它不会崩溃，或者也只是很偶然地崩溃。
2. 传输延迟为零。信息从发送端发出的同时接收端就会收到。
3. 带宽是无限的。通过网络，您可以随心所欲地发送大量信息。这个谬误的推论是说带宽不会导致传输延迟或者影响网络可靠性。
4. 网络是安全的。没有人可以攻击网络、扰乱数据流、窃取数据等。
5. 网络拓扑（网络布局）不会改变。任何一台电脑一旦存在于网络上，将一直保留在网络上。
6. 网络有一个管理员。网络的所有变化只能由这个管理员做出，问题也累积到这个管理员那里。这个管理员是绝对可信的，不可能犯错误。
7. 传输成本为零。因此，您可以尽您所想地发送信息而无需支付费用。
8. 最后，网络都是相似的。它使用同样的硬件、操作系统、安全工具。网络基础设施的配置也是一样的。

我想，即便您不是一个专业技术人员，也能从日常生活中发现以上论述中至少一两点是错误的。例如，试想每天通过网络传输的大量垃圾信息。传输成本显然不是零，或者说人们故意当作无成本罢了。

pin51.com 新书发布 资源分享

6.6 Wiegers 检查表评审练习

在下面的练习中，请将 Wiegers 的需求评审检查表和用例评审检查表运用到 HELLOCARMS 系统需求规格说明中。

练习由 3 部分组成：

1. 准备：使用 Wiegers 检查表去评审 HELLOCARMS 系统需求规格说明（附录 B 010 部分），并记录您发现的问题。

2. 评审会议：如果您正在使用本书，那么可以在一个小组中来进行走查，创建出一个问题清单。

3. 讨论：在进行了走查后，与其它小组和指导老师讨论你们的发现。

第一部分的解决方案将在下一章节中说明。

6.7 Wiegers 检查表评审练习解答

RBCS 的资深合伙人 Jose Mata 运用 Wiegers 检查表评审了 HELLOCARM 系统需求规格说明，提供了如下反馈：

- 内部的交叉式引用是否正确？如果我们在这个文档中引用另一个文档或部分，是否这个引用有效？这是很容易出现的错误，并且一旦出现将会令人非常迷惑。
 - 没有。010-010-040 部分陈述，“输入域级验证的细节有一个单独的文件描述”，但是这个文档没有在需求文档的任何地方进行标识。
- 细节的详细程度是否一致并准确？
 - 没有。010-010-180 部分陈述：“提供各种功能和屏幕，支持 Globobank 的小额分行的运作”。这样的说明太含糊了而不能被接受。
 - 010-010-170 部分陈述，“支持通过互联网递交申请，并保证未经训练的用户也能正确地进行申请操作”，这是一个非常大而模糊的需求。
 - 010-010-190 部分到 010-010-240 部分表达为，“支持房屋净值申请的市场交易、销售和处理……”，这是非常模棱两可的，以致一些重要的功能可能被忽略。
- 需求是否为设计提供了足够的数据？
 - 没有。010-010-070 部分陈述，“询问每个申请人当前是否与 Globobank 有业务往来，例如，拥有支票或储蓄账户”，但是这个说法并不完整，尽管它本应该表达完整。
 - 010-010-080 部分陈述，“维护申请状态信息，从开始到反对、拒绝，或接受……”，但是我们不知道这些状态是否是一个子集还是全面的。
 - 010-010-150 部分陈述，“对于 Globobank 运营覆盖到的所有州、省和国家，提供本地和远程的电话销售支持”，而这个列表并没有被定义。
 - 010-010-160 部分陈述，“提供限定的合作伙伴专用的屏幕、标识、界面和品

pin51.com 最新最全 拼吾爱

牌，以支持代理商和其他业务伙伴”，但是显示屏或接口的范围并没有定义。
 - 010-010-250 部分陈述，“支持灵活的定价策略，包括建议价格，短期价格和其他方式”，但是“其他”代表什么需要定义。
- 是否包括需求的优先级？
 - 是的。
- 是否定义了外部接口？
 - 没有。我们不知道信息的全面性如何。数据结构被暗示，可是没有进行定义。暗示的接口如下所示：
 - LoDoPS: 010-010-050, 010-010-100, 010-020-050, 010-030-040, 010-030-060, 010-030-070,010-030-080,010-030-120,010-030-103,010-030-140,010-030-150
 - GLADS: 010-010-070
 - 评分主机: 010-020-020, 010-030-020
 - GloboRainBQW: 010-030-010
- 是否存在一些缺失的信息？如果是这样，是否明确地标记了 TBD“待定义”？
 - 是的。
- 是否记录了预期的行为？
 - 没有。例如，010-010-080 部分陈述，“维护申请状态信息，从开始到反对、拒绝……”，但是并没有说明如何、在哪儿维护信息。
- 是否每个需求表达清楚、简洁，不含糊？
 - 没有。例如，010-010-070 部分陈述，“询问每个申请人当前是否与 Globobank 有业务往来”，但是我们并不清楚如何询问申请人。
 - 010-010-100，010-030-040 和 010-030-070 部分陈述，“允许用户在一个单独的屏幕上说明客户将要收回的债务”，但是我们并不清楚是将什么样的显示屏独立分开。
 - 010-040-010 部分陈述，“支持商定的安全要求（加密、防火墙等）”，是很模糊的。
 - 010-040-060 部分陈述，“对所有金融申请的处理，支持欺诈侦测”，这是非常模糊的，尤其对于优先级为 1 级别的需求会非常糟糕。
- 是否每个需求可验证？是否可以设计一个测试用例去证明需求被实现？
 - 没有。010-010-150 部分陈述，“提供本地和远程的电话销售支持……”，这是很不清晰的，也无从得到验证。
 - 从 010-010-190 到 010-010-240 部分陈述为，“支持市场，销售，并处理……”，而市场部分不可验证。
 - 010-010-250 部分陈述，“支持灵活的定价策略，包括建议价格，短期价格，以及其他”，但是“其他”不可验证。
 - 010-030-150 部分陈述，“支持计算机电话集成，为本地电话促销活动和品牌业务伙伴提供个性化的市场和销售支持”，这是很不清晰的，也无从得到验证。
- 是否每个需求都在范围之内？

ptn51.com 新客发 新客服务 新客福利

- 没有。010-010-170 部分陈述，“支持通过互联网递交申请，并保证未经训练的用户也能正确地进行申请操作”，这个需求超出了 003 部分，因为网络用户是在后续版本中才会允许的。
- 010-040-030 部分陈述，“允许外包电话推销员查看信贷级别，但不能查看申请人的实际信用评分”，这个需求超出了 003 部分。
- 010-040-050 部分陈述，“允许互联网用户浏览潜在贷款，直至申请过程中的最新可行点。这些用户无需透露姓名、政府的识别号码等个人信息”。这个需求超出了 003 部分。

■ 是否每个需求没有内容和语法错误？
- 是的。

■ 需求能够在范围里实现吗？
- 不一定。010-040-060 部分陈述，“对所有金融申请的处理，支持欺诈侦测”。这一步不一定能被实现。特定的检查需要被定义。

■ 是否所有的安全因素进行了恰当地说明？
- 没有。特定类型的用户和他们的权限未定义。未提及用户名和密码长度。未提及特殊数据的加密。维护和清理需要没有被提及。服务器物理安全需求没有被提及。

■ 是否唯一并正确地标识每个需求？如果需求范围不明确，将不可能从测试中回溯到需求。
- 没有。例如，从 010-010-190 到 010-010-240 部分陈述为，“支持市场、销售、并处理……”，这个需求粒度太大了。
- 010-010-250 部分陈述，“支持灵活的定价策略，包括建议价格、短期价格，以及其他”，如果将这个和其他的复合需求分离开来并作为一个单一的需求，那么将会更加清晰。也许需求在一定程度上稍有重复，但是需求更加明确，并能更好地平衡开发和测试的花费。

■ 是否我们处在需求的王国里，而不是设计？换言之，是否所有的需求确实是需求而不是设计或者实施解决方案？
- 没有。010-010-100, 010-030-040 和 010-030-070 部分陈述，“允许用户在一个单独的屏幕上说明客户将要收回的债务，如果有……”，详细说明一个独立的显示屏似乎是设计的细节。

6.8 Deutsch 检查表评审练习

在 HELLOCARMS 系统需求规格说明中的表格开头部分，可以得出它是分布式的。事实上，它是高度分布式的，因为多重网络连接必须为它从应用到功能进行服务。

在这份练习里，请将 Deutsch 的分布式应用设计评审检查表运用到 HELLOCARMS 系统需求规格说明中。

练习包括 3 部分：

pin51.com 最新最全源码分享 拼吾爱

1. 准备：基于 Deutsch 的检查表，评审 HELLOCARMS 系统需求规格说明（只针对 010 部分），识别潜在的设计问题。

2. 评审会：假设您正和其他人一起工作，在一个小型的组织里开展走查，创建出一个问题清单。

3. 讨论：在进行走查后，与其他的小组和指导老师讨论你们的发现。

第一部分的解决方案将在下一节中说明。

6.9 Deutsch 检查表评审练习解答

RBCS 的资深合伙人 Jose Mata 运用 Deutsch 检查表来评审 HELLOCARMS 系统需求规格说明，提供了如下反馈：

- 网络是可靠的，它不会崩溃，或者也只是很偶然地崩溃。
 - 附录 B 中图 1 HELLOCARMS 设计不包含备份方面的考虑，比如崩溃的服务器、后备开关、备选网络路径。当失去通讯连接时，保证系统不丢失信息需要的费用没有指出。
- 传输延迟为零。信息从发送端发出的同时接收端就会收到。
 - 效率不在 010 需求范围之内。
- 带宽是无限的。通过网络，您可以随心所欲地发送大量信息。
 - 效率不在 010 需求范围之内。
- 网络是安全的。没有人可以攻击它，扰乱数据流、窃取数据等。
 - 010-040-010 部分没有包括让人对该系统安全性充满信心的细节。
- 网络拓扑（网络布局）不会改变。任何一台电脑一旦存在于网络上，将一直保留在网络上。
 - 020 部分中的容错性和可恢复性待定义。关于应用将包括在 010-010-060 部分中。
- 网络有一个管理员。网络的所有变化只能由这个管理员做出，问题也累积到这个管理员那里。这个管理员是绝对可信的，不可能犯错误。
 - 特定类型的用户和他们的权限，没有被定义。
- 传输成本为零。因此，您可以随心所欲地发送信息而无需支付费用。
 - 010-010-160 部分没有范围限制，“提供限定的合作伙伴专用的屏幕、标识、界面和品牌，以支持代理商和其他业务伙伴”，如果空着未定义，一些图形可能会很大。
- 最后，网络都是相似的。它使用同样的硬件、操作系统、安全工具。网络基础设施的配置也是一样的。
 - 所支持的电脑系统、操作系统、浏览器、通信协议等，未定义。应对版本进行详细说明，尽管这个细节或许应该包含在一份设计文档中而不是需求规格说明中。

6.10 认证考试模拟题

在每个章节的最后，您可以试着做几道认证考试模拟题以加强对材料的记忆和理解，准备 ISTQB 高阶测试分析师的考试。

1. 下面哪种类型的评审能最有效、最大限度地发现评审对象中的缺陷？

A 非正式的　　B 走查

C 同行评审　　D 审查

2. 一份设计规格说明书陈述如下：

“使用 TCP/IP 的 10MB 及以上的网络连接为数据服务器和应用服务器提供了接口”。

假设被测系统需要在不到 1 分钟之内传输达到 1GB 数据量。

下面哪项陈述最形象地描述了由这个状态导致的可能结果？

A 系统将遭受可用性问题　　B 系统将遭受性能问题

C 系统将遭受维护问题　　D 这种状态不会导致任何问题发生

3. 假设一个测试分析师在为银行项目工作，他正在升级一台 ATM 机以便客户可以用所支持的信用卡来预借现金。

“对于银行所支持的信用卡，系统允许至少 500 美元的取现，银行支持的信用卡有 American Express、Visa、Japan Credit Bank、Eurocard 和 MasterCard”。

下面的哪一项描述是正确的？

A 对于所支持的信用卡，这段陈述是有歧义的

B 这个场景隐含着潜在的性能问题

C 对于取现的额度范围并没有明确规定

D 这个场景隐含着潜在的可用性问题

4. 下列哪种评审类型涉及合规性判断？

A 审计　　B 走查

C 审查　　D 管理评审

pin51.com 拼吾爱

第7章

事件管理

> “更快/随机。”
>
> 作者和他的同事在一特定项目的一份文字有点玄妙的完整的缺陷报告中所观察到的。

高级大纲的第 7 章是事件管理。任何时候，只要实际结果和预期结果不同，都称为一个事件的发生。高级大纲里采用的是 IEEE 1044 标准，主要讲的是事件的生命周期和事件报告需要的信息，这些信息应该由测试人员搜集。高级大纲的第 7 章包括 6 小节内容：

1. 概述。
2. 何时可以发现一个缺陷。
3. 缺陷生命周期。
4. 缺陷要素。
5. 度量和事件管理。
6. 事件沟通。

下面我们来看其中每一小节，以及它们与测试管理的关联。

7.1 概　　述

学习目标：

能回想起该部分内容即可。

事件管理是所有测试人员必备的技能。测试经理更关注测试过程。从事件识别到调查，到处理，到处置，必须要顺畅、按时地前进。测试人员要准确记录事件，并在事件处理过程中，进行适当的确认测试和回归测试。

不同角色的测试人员工作的重心也有所不同。测试分析师从商务和用户需求角度，比较实际和预期结果的差异。技术测试分析师则要评估软件本身的行为，甚至可能需要具备更深入的技术和见解。

pin51.com 新考点 资源分享平台

7.2 何时可以发现一个缺陷

学习目标:

能回想起该部分内容即可。

一旦有了草拟的需求规格说明，我们就可以开始静态测试以发现缺陷。只要有了可执行组件，我们就可以开始动态测试，发现作为缺陷征兆的失效。

测试好比过滤活动。为了达到可能的最高质量，在整个软件生命周期，我们应该进行静态和动态的测试。如果采取大量的早期过滤活动，比如需求评审、设计评审、代码评审、代码分析等，除了滤出缺陷外，我们还可以进行早期缺陷检测和清除。这样做很可能降低总体成本，同时也能降低进度安排失误带来的风险。

在发现失效时，我们不应该立即断定此失效是被测系统的一个缺陷。因为测试用例也有可能存在缺陷。

7.3 缺陷生命周期

学习目标:

（K4）在可理解的缺陷报告中对功能性/非功能性缺陷进行分析、分类、描述。

图 7-1 表示 IEEE 1044 事件管理生命周期，包含了事件跟踪系统中的事件报告。下面来学习缺陷生命周期。

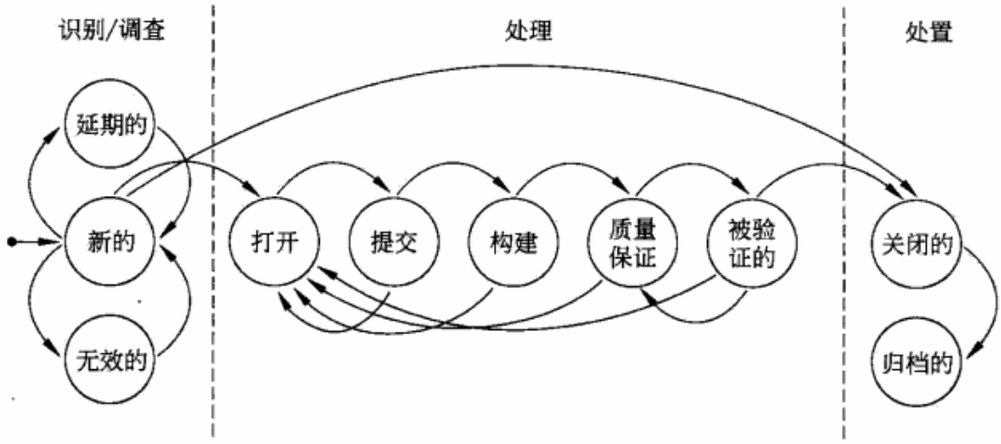


图 7-1 IEEE 1044 事件管理生命周期活动

ISTQB 术语

缺陷（或者 bug）：可能会导致软件组件或系统无法执行其定义的功能的瑕疵，例如：错误的语句或变量定义。如果组件或系统在运行中遇到缺陷，可能会导致运行的失败。

pin51.com 最新最全源码素材 拼吾爱

错误：人为地产生不正确结果的行动。

失效：组件或系统与预期的交付、服务或结果存在的偏差。

事件：任何有必要调查的事情。

事件日志：记录所发生的（例如，在测试过程中）事件的详细情况。

根本原因分析：以发现缺陷的根本原因为目的的分析技术。通过直接纠正根本原因，减小缺陷的发生率。

ISTQB 术语

异常：任何和基于需求文档、设计文档、用户文档、标准或者个人的期望和预期之间偏差的情况，都可以称为异常。异常可以在但不限于下面的过程中识别：评审、测试分析、编译、软件产品或应用文档的使用等。

假设在每个事件的生命周期，从最初的识别到最终的处置，其不同状态是按照一定的顺序出现的。不过，从图 7-1 也可以看出，并非所有事件经历严格相同的状态变化过程。IEEE 1044 缺陷生命周期包括 4 个阶段：

- 阶段 1：识别。发现某个异常，并且该异常属于潜在的缺陷。软件生命周期的任何阶段都可能发现缺陷。
- 阶段 2：调查。在识别之后，进入对潜在缺陷的调查阶段。调查能够揭示相关问题，也能促进问题的解决。调查之后可能得出这样的结论：事件不是由实际的缺陷引发的，可能是测试数据存在问题。
- 阶段 3：处理。调查结果决定下一步的行动，可能是采取措施处理缺陷，也可能是预防将来发生类似的缺陷。在缺陷排除之后，必须做回归测试和确认测试。任何被缺陷阻塞了的测试，这时候都可以重新开展了。
- 阶段 4：处置。处理结束后，事件进入处置阶段。这个阶段原则上是捕捉进一步的信息，并使事件进入终止状态。

当然，推进事件从一个状态进入另一个状态，也即从生命周期的一个阶段进入另一个阶段的是我们对事件的不断理解。之所以定义不同的状态，是因为缺陷会从一个人手上递交到另一个人手上，我们必须捕捉这些信息。事实上，在每个阶段，更确切地说，每个状态，包括 3 种信息捕捉活动：

- 记录。
- 分类。
- 评估影响。

这个过程的实施方法如表 7-1 所示。

在识别阶段，我们要记录有说服力的数据，基于已经观察到的重要特征进行分类，也要根据已发现的影响判断事件将会产生的影响，尽管此时的判断可能与最终评估有差异。

表 7-1 IEEE 1044 过程分类

阶 段	活 动		
	记 录	分 类	评估影响
1. 识别	支持的数据	根据重要特征	根据已察觉的影响
2. 调查	更新和添加支持数据	更新和增加重要特征的分类	基于调查进行更新
3. 处理	增加处理相关的数据	增加处理相关的数据	基于处理更新
4. 处置	增加处置相关的数据	基于处置	根据处置更新

在调查阶段，我们要更新并记录更多支持的数据，同时，根据此阶段发现的新的重要特征，也应该更新已有的并增加新的分类。此外，我们还要进一步评估事件的影响。

在处理阶段，根据所采取的行动，我们要记录新的支持数据、增加分类和更新事件的影响。

最后，在处置阶段，我们要记录最终的数据，调整并确定如何分类，完成事件影响的评估。

注意，我曾提到过支持数据及分类，IEEE 1044 标准包括每个阶段每个活动的必要的和可选的支持数据及分类。在下面几小节中，我们还会复习这些内容。

所有这些数据项和分类都与某个阶段或活动有关。IEEE 标准中使用两个字母的代码来表示这些词：RR（识别），IV（调查），AC（处理），IM（评估影响）以及 DP（处置）。

我们会一一描述这些阶段的数据项和分类。但是，不要只见树木而忽视了森林。重要的不是思考“我的事件管理系统是否符合 IEEE 1044 标准？”，而是去想清楚“这些数据或者分类方法是否有意义？”。

识别阶段的分类如下：

- 项目活动（RR1XX）：发现事件时您在做什么？这是 IEEE 1044 中必要的数据。
- 项目阶段（RR2XX）：项目处于什么阶段（必要）？这要根据项目生命周期的具体情况来回答。
- 可疑因素（RR3XX）：您认为什么是可能的原因（可选）？我发现，许多情况下，捕捉这些数据对开发人员很有帮助，特别是对那些十分技术型的测试人员。
- 事件重现（RR4XX）：可否让事件再出现一次（可选）？IEEE 1044 中这项是可选的，但是我认为要做事件报告，能够让事件复现是非常必要的。
- 事件症状（RR5XX）：事件本身是如何暴露自己的（必要）？
- 产品状态（RR6XX）：如果事件不处理，会对产品使用造成什么影响（可选）？我也不同意 IEEE 1044 将此列为可选的，这应该也是有必要分析和收集的信息。

标号最后的 XX 代表层次，即每个分类又可以包括子类。比如，项目活动类的标号有 RR110，RR120 等。IEEE 1044 中对各个子类也有定义，在此，我们不再回顾这些子类的细节。

识别阶段的数据如下：

- 您在发现事件时所处的工作环境是怎样的？对此，您应该记录产品硬件、产品软件、数据库、测试支持软件、平台、固件和其他有用信息。

pin51.com 最新最全 拼吾爱

- 最初的详细情况是怎样的（包括当时的测试人员是谁）？您应该记录您的名字，发现事件的日期，代码或者功能范围，测试对象的版本，还有其它联系信息，比如电子邮箱、通讯地址、电话号码和公司 ID 号。
- 您是在什么时间察觉事件的？包括操作时间（从上次重启系统算起或者就是整个系统运行时间）、挂钟时间、系统时间和 CPU 时间。
- 相关的供应商信息有哪些？包括公司名称、联系人姓名、供应商 ID、预期解决方案和预期的解决日期。

调查阶段的分类如下：

- 实际原因（IV1XX）：是什么原因真正导致了事件的发生（必要）？
- 根源（IV2XX）：事件的根源是什么（必要）？这个问题包括潜在的错误。
- 类型（IV3XX）：什么类型的缺陷导致了失败（必要）？这是一个缺陷分类问题。

记住，在此阶段可以更新前一阶段的分类和数据。

调查阶段的数据如下：

- 我们能获取什么有效信息？收到什么数据？报告编号是多少？谁是调查人员？估计调查的起止日期？实际调查的起止日期？调查用了多少人和多少小时?我们得到调查结果确认的日期？在调查中使用了什么文档？
- 我们能获取什么验证信息？异常（或者事件）的起源是什么？我们如何验证识别过程提供的数据？

记住，在此阶段可以更新前一阶段的分类和数据。

处理阶段的分类如下：

- 解决方案（AC1XX）：何时把事件处理完，用什么方法（必要）？
- 矫正行动（AC2XX）：怎样防止未来发生类似事件（可选）？

记住，在此阶段可以更新前一阶段的分类和数据。

处理阶段的数据如下：

- 我们应该捕获什么信息以识别解决方案？要修正的测试项是什么，特别要修正其中什么要素？如何以文本方式描述处理过程？计划完成处置的日期？此任务分派给谁？计划修正问题的日期？如修正被延缓，授权人是谁？依据是什么？
- 我们能捕获什么方案行动的信息？完成处理日期？解决方案由什么组织和谁来最后验证？

记住，前一阶段的数据，在此阶段可以更新。

下面是处置阶段的分类和支持数据：

- 处置（DP1XX）：问题最终是如何被解决的（必要）？
- 我们要获得哪些异常（或事件）处置信息？采取了什么行动？报告什么时候关闭？什么时候文档更新完成？什么时候通知的客户？可能存在的参考文档编号有哪些？
- 我们要记录什么验证信息？执行验证的人的姓名？验证发生时的日期？验证的版本和修订级别是什么？验证采用了什么方法？验证时用的测试用例是什么？

记住，在此阶段可以更新前一阶段的分类和数据。

pin51.com 最新最全源码资源下载

ISTQB 术语表

优先级：赋予某项（业务）重要性的级别，如，缺陷。

严重性：缺陷对组件/系统的开发或运行造成的影响程度。

贯穿事件的整个生命周期，影响的类别也会经过不断分类和修正。下面来看其中一些分类：

- 严重性（IM1XX）：对系统的影响是什么（必要）？
- 优先级（IM2XX）：事件的相对重要性（可选）？我不赞同这项可选，在很多情况下，优先级比严重性更重要。
- 客户价值（IM3XX）：事件如何影响客户或者市场价值（可选）？我认为此项也很必要。
- 任务安全性（IM4XX）：事件是否影响任务的目标实现或者安全性（可选）？当然这项只针对特定系统。
- 项目进度（IM5XX）：处理这个事件对项目进度的影响是什么（必要）？
- 项目成本（IM6XX）：处理这个事件对项目成本的影响是什么（必要）？
- 项目风险（IM7XX）：处理这个事件是否带来项目风险（可选）？此项我觉得也是必要的。
- 项目质量/可靠性（IM8XX）：处理这个事件对项目质量或者可靠性带来的影响是什么（可选）？这也应该是重要因素。
- 社会影响（IM9XX）：处理这个事件是否带来社会影响（可选）？这项只针对某些系统。

事件影响的数据如下：

- 事件对成本的影响，包括分析的成本、问题成功修正的估计成本、问题修正失败的估计成本，以及事件解决带来的其他成本。
- 事件对时间的影响，包括问题成功修正的估计时间，问题修正成功的验证估计要花多少时间，问题修正失败的估计时间，实际执行时间。
- 事件所带来的风险，需要文本描述。
- 事件对进度的影响，包括假设事件被处理了，处理失败。如果处理成功，对进度的实际影响。
- 事件导致合同的任何变更。

同样记住，在生命周期的后期，如果有必要的话，这些数据项还是可以修改的。

7.4 缺陷要素

学习目标：

（K4）在可理解的缺陷报告中对功能性/非功能性缺陷进行分析、分类、描述。

pin51.com 最新拼吾爱

前面已经了解了 IEEE 1044 中缺陷要素的定义。本小节介绍如何应用 IEEE 1044，特别是怎样将 IEEE 1044 定义的生命周期和要素应用到您的事件管理中，比如，前面一节中的必要和可选的分类和数据。在事件报告的整个生命周期，报告的撰写者会不断设置和更新这些要素，尤其是在事件状态转变时。图 7-1 显示了 IEEE 1044 如何将生命周期映射成典型的事件报告状态转换图。

IEEE 1044 的姊妹标准 IEEE 1044.1，说明了如何在您的组织中实施一个符合 IEEE 1044 的事件管理系统。IEEE 1044.1 的作者了解到不同组织对其事件分类和数据的命名会不同。于是，IEEE 1044.1 中定义了一个过程，用于将 IEEE 的事件管理术语映射到特定组织使用的名称。无需重新命名分类和数据，也不用更改事件生命周期，您的组织所用的事件管理系统就可以实现与 IEEE 1044 的合规性。

为什么要遵照 IEEE 1044？毫无疑问，在您的公司不同的项目使用一致的事件管理过程是有意义的。我曾经试图为一个客户比较两个软件的质量，一个是当前项目的软件，另一个是过去项目的类似软件。由于事件管理过程不一样——不仅是工具不同（我可以处理），而且记录的分类和数据的含义也不同，质量比较无法进行。如果您整个公司的事件管理系统符合 IEEE 1044，那么您不仅可以比较公司内不同项目，还可以与其他遵照 IEEE 1044 的公司进行比较。

将 IEEE 1044 标准应用到您的组织中，过程如下：

- 步骤 1：将您当前的分类与 IEEE 1044 分类进行比对。这会让您了解相关活动的工作量。
- 步骤 2：评估遵照 IEEE 1044 需要多大工作投入，决定要不要或者多大程度上遵照 IEEE 1044。记住，没有必要完全一致，或者完全不一致。您可以决定在某些方面、某些类别、某些数据项遵照标准，其他的可以不一致。
- 步骤 3：不断回顾 IEEE 1044 的分类。这样做是考虑到，IEEE 1044 中有的分类方式很容易整理，对当前事件的处理有用，或者可用于将来做分析。可以把这些分类应用于您的系统。
- 步骤 4：选择关键的分类并实施。
- 步骤 5：定义如何使用这些分类信息。例如，使用这些事件数据做哪种类型的分析，什么时候进行分析？这点很重要，因为在事件管理过程中您必须先做相关设置，使得搜集的数据对分析有用，并且能及时提供这些数据。
- 步骤 6：记录与 IEEE 1044 分类对应的类别。例如，哪个是识别阶段的分类，哪个是事件影响的分类等。当然，在分类命名上，与 IEEE 1044 标准的差异越小，记录越容易。
- 步骤 7：记录所有分类及其用途。
- 步骤 8：如果您关心与 IEEE 1044 的合规性，那么您就有必要以文档方式记录与 IEEE 1044 一致和不一致的地方。这样做有什么意义？如果您的事件管理系统只是自己使用，可能就没有意义。但是，如果您是一个软件或者测试服务公司，并

且您打算将您的事件管理系统和客户的联系起来，那么 IEEE 合规性可以作为您们共同的参考点。

- 步骤 9：定义每个阶段要搜集的支持数据。
- 步骤 10：记录需要搜集的支持数据。在 IEEE 1044.1 标准中把步骤 9 和步骤 10 放在这里，我个人认为这两个步骤应该和步骤 1 和步骤 5 并行做。
- 步骤 11：映射各个状态的分类和数据到您当前的事件、问题或者缺陷跟踪系统中。IEEE 认为系统正确的名称是事件，因为只有调查之后您才能知道发生的行为是否为缺陷。不过，怎么命名还是按照您的意愿吧！
- 步骤 12：确定搜集分类和数据的过程并且以文档形式记录。换句话说，某个分类或者数据项最初属于哪个状态？到什么状态时它们被更新了？在什么状态它们必须被更新？谁可能或者必须输入初始信息？谁可能更新信息？谁必须更新信息？
- 步骤 13：信息使用计划。我认为这个步骤应该和步骤 2 并行做，放在这里有点晚了。
- 步骤 14：培训使用者和管理者。使用者应该懂得如何输入和更新分类信息和数据。管理者需要知道如何使用这些信息。

所以，后面这几小节传达了如何基于 IEEE 1044 合规性来扩展您的事件管理系统。但是，IEEE 1044 合规性只是对待事件的一种手段。我们真正要实现的是什么呢？

记住，我们捕捉所有信息是为了用它们来做些事情。我们立即要做的是采取行动处理事件。事件报告应该记录可采取行动的信息，具备如下属性：

- 完整。不能错过任何重要细节。
- 简洁。不要围绕细枝末节说个不停。
- 精确。不会误导读者，也不误报。
- 客观。尽可能多地基于事实，不对任何开发人员抱有偏见。

除了要针对单个事件报告采取行动外，不要忘记许多测试度量元是从对事件报告的总体分析推导出来的。基础级大纲中管理测试一章讨论了这一点。所以，捕捉精确的分类信息，风险分析信息和过程改进信息对于事件报告是很重要的。

如图 7-2 所示案例，在一个项目中使用分类信息来学习一些有价值的东西。在一个大型复杂的项目中，Pareto 图表分析每一个主要的子系统——系统中的缺陷数和百分比。我们称这个项目为 NOP 项目，它包含 10 个子系统，通过广域网、局域网和电话系统来实现一个分布式的大型娱乐系统的运作。

在图 7-2 中，互动语言响应（IVR）应用程序包括了几乎一半的缺陷，客户服务应用程序（CSA）包括了 30%左右的缺陷，其余的应用程序相对来说缺陷比较少。内容管理（CM）应用程序发现的缺陷占所有发现缺陷不到 10%。互动语音响应应用程序服务器的电话层和 OS/硬件层分别占据了约 5%，其余的应用程序和结果占据了剩余的 4%。

pin51.com 最新最全源码分享下载站 拼吾爱

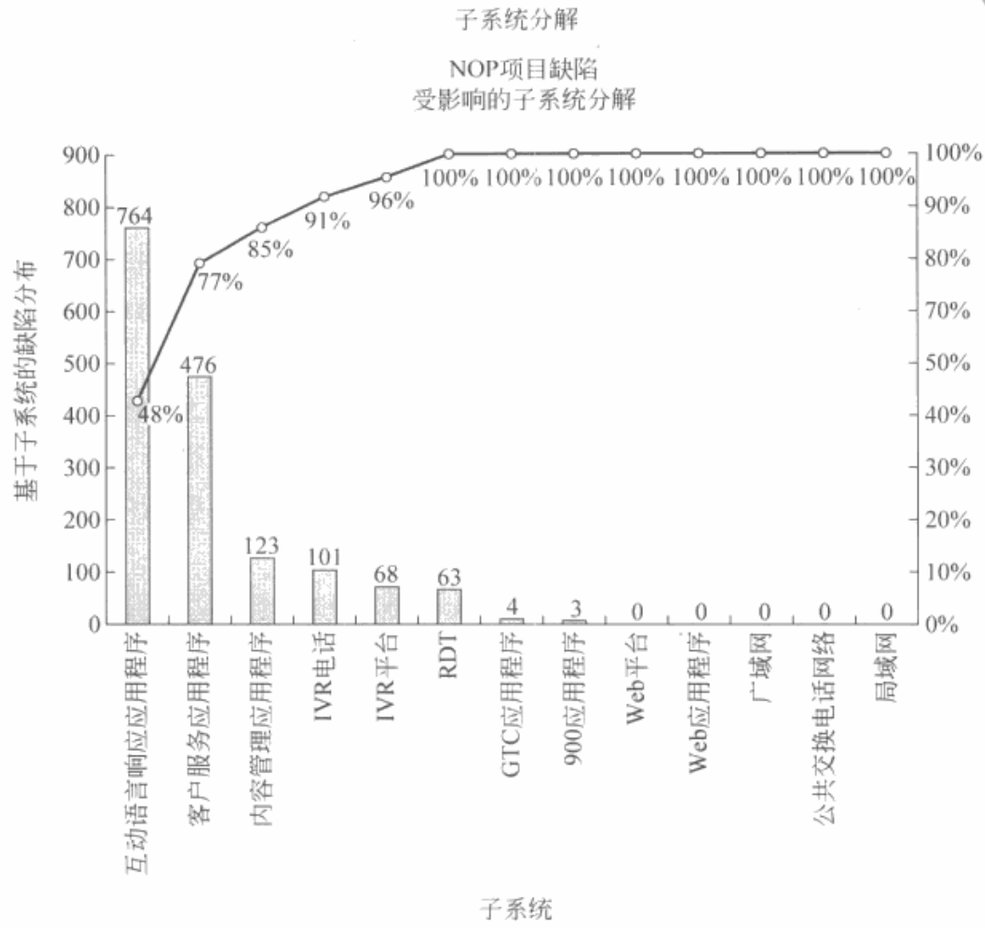


图 7-2　基于子系统的缺陷分类

7.5　度量元和事件管理

学习目标:

能回想起该部分内容即可。

对于测试分析师来说，事件报告中的文本描述信息的用处是显而易见的，但是对于分类的使用却很容易迷惑。我的一个客户告诉我他们在咨询服务上花费了几十万美元来改进事件跟踪系统，目的是为了使用最新的分类方法，即正交缺陷分类法。但他们却在人员培训上竭力节省开销，这使得所有的分类信息变得毫无价值可言。

我们需要从项目的角度和长远的组织上关于规程改进的角度来对待事件分类信息。从项目的角度来看，事件分类应该是辅助基础级大纲中讨论的测试规程监控。不同的度量元如缺陷群分析、缺陷密度分析、缺陷聚合（也被称作打开/关闭图表）都可用于项目中来管理缺陷趋势并检查（产品）是否可以发布。

从组织和过程的角度来看，我们可以去评估我们是如何做的，并去设想如何才能做得更好。因此，事件分类应该支持过程改进活动。我们应该能够通过评估在同一阶段分别有多少百分比的缺陷被检测和移除来对该阶段进行控制。我们应该能够评估根本原因

pin51.com 新客发 新客免费资源分享

以减少总体的缺陷数。最后，我们还应该能够贯穿整个项目的缺陷趋势，分析最佳和最差实践部分。

7.6 事 件 沟 通

学习目标:

能回想起该部分内容即可。

一份糟糕的事件报告是导致项目团队中矛盾和不和谐关系的主要原因。我的同事和我一直碰到这样的问题。记住如下几点将有助于保持团队中的良好关系。

谴责或把错误归咎于某人通常不是测试人员的工作，尽可能避免那些带有责备的评论和带个人感情色彩的评论。

一份好的事件报告应该提供客观的信息，着眼于事实。如果您将做一个假设或陈述一个理论，请先说明之所以这样做的理由。如果您决定去进行这样的假设或理论，确保要记住第一原则“不要个人主义”。

事件报告通常是断定某些方面是错误的。因此，当您在提出存在一个问题时，其实是向正确迈了一步。尽力追求最大化的准确度。

最后，这更多是有关思想意识但是也很重要的一个认识——将事件报告当作您为经理和开发人员提供的一项服务。去询问开发人员您的报告中应该包含什么样的信息来帮助他们，您会非常惊奇这样做所创造的不同。

当“他们”提交的缺陷没有得到修正时，一些测试人员往往感到沮丧。当我们在评估过程中发现这一点时，我的第一想法就是事件管理过程出现了一些问题。理想情况下，一个缺陷诊断或者缺陷会诊会议举行时，需要包括跨部门的利益相关者，共同去判定事件的优先级。仅仅是听取开发人员或测试人员关于什么应该修正或延期的意见是不可取的。这并不是说开发人员和测试人员的意见和想法没有价值，而是因为良好的事件管理要求对处理事件的方式进行慎重考虑。没有几个项目可以真正地做到修正每一个出现的事件。

总之，团队中有效的沟通和良好的关系，良好的缺陷跟踪工具和缺陷会诊会议对于创建良好的事件管理规程非常重要。事件管理是所有测试分析师都应掌握的基础知识。

ISTQB 术语

配置控制委员会：负责评估、批准或拒绝配置项修改的组织，此组织应确保被批准的配置修改的执行。

7.7 事件管理练习

假设选定一组电话银行客户经理参加 HELLOCARMS 系统的 Beta 测试。银行客户经理将以客户身份进入在线申请系统。他们随后也会选取信息并输入到当前系统中，以确

pin51.com 最新最全 拼吾爱

保 HELLOCARMS 没有缺陷对面向客户的功能造成影响。这些银行客户经理并不是训练有素的测试人员，并且也不乐意花时间去学习测试的基础知识。因此，为了避免银行客户经理书写糟糕的事件报告，并给事件报告的度量元引入噪声，造成不良影响，管理层决定，当银行客户经理发现问题时，他们即刻给测试分析师发邮件并由测试分析师来书写报告。

您收到了一封银行客户经理发来的邮件并描述了如下的问题：

“我在为一个信用良好的客户输入房产贷款申请，她拥有高价值的房产，然而贷款额并不是很大。”

HELLOCARMS 在显示屏上显示“升级到高级电话银行客户经理”的信息。然后我单击了继续按钮。但是在我没有输入高级电话银行客户经理授权码的情况下，系统竟然允许执行下一步操作。

这一步骤之后，客户申请操作运行正常。

我的另一个客户申请条件相似——高价值房产，中等贷款额——当天晚些时候也碰到此种情形。在没有输入授权码的情况下我同样得以继续操作。

这份练习包括两个部分：

1. 在这封邮件中，有哪些是符合 IEEE 1044 识别和识别影响分类和数据？
2. 您将采取什么措施来说明这份报告？

答案在下面说明。

7.8 事件管理练习解答

首先，我来评估每一个相关的识别和识别影响分类和数据范围去查找我假设的关于这封邮件和其他信息是否都具备，如表 7-2 所示。

表 7-2 IEEE 1044 事件报告范围

IEEE 信息	可行？
项目活动	假定这一步是为了进行 beta 测试
项目阶段	假定这一步是为了进行 beta 测试
可疑原因	不明
重现性	明确，但是需要更多地分离和重现
症状	明确
产品状态	不明，但是我们可以假设产品不接受电话银行客户经理去绕过风险管理政策
环境	假定这一步是为了进行 beta 测试
发起人	假定包括在发送者的邮件中
时间	不明
供应商	我们可以假设知道一些供应商的信息，但是有些信息并不知道，如不知道他们将何时修正缺陷
严重度	明确
优先级	不明，但是我们可以假设这为高优先级

续表

客户价值	不明，但是可推断
任务安全性	不适用
项目进度	由于需要进行调查，因此目前不适用
项目成本	由于需要进行调查，因此目前不适用
项目风险	由于需要进行调查，因此目前不适用
项目质量/可靠性	由于需要进行调查，因此目前不适用
社会影响	由于需要进行调查，因此目前不适用
成本	由于需要进行调查，因此目前不适用
时间	由于需要进行调查，因此目前不适用
风险	不明，但是我们可以做一些推断，并描述让电话银行客户经理绕过风险管理政策而带来的风险
进度	由于需要进行调查，因此目前不适用
合同	由于需要进行调查，因此目前不适用

接下来，我在输入系统之前对报告中拟定阐明的一些步骤进行了注释。原始的信息是用斜体表示的，我的文本是用正常字体表示的。

我在为一个信用良好的客户输入房产贷款申请。

我想查出有关这个客户的确切数据，包括收入、债务、财产、信用度等。

她拥有高额的房产然而贷款额度并不是很大。

我想查出房产和贷款额的确切价值。

我将尝试一些值和贷款额度的不同组合来看看我是否能找到一些规律。

HELLOCARMS 在显示屏上显示“升级到高级电话银行客户经理”的信息。然后我单击了继续按钮。但是在我没有输入高级电话银行客户经理授权码的情况下，系统竟然允许执行下一步操作。

我会查出银行人员是否输入了信息。

我会测试如下的一些步骤，如不输入任何字符、输入空白、输入不是授权码的有效字符，并且进行一些其他的检查，最终得出这个选项是否完全被忽略。

从前面提及的客户申请来看，一切操作运行正常。

我会尝试这些申请是否被转移到 LoDoPS 还是被悄悄地取消了？如果被转移到 LoDoPS，LoDoPS 是否会做相应处理，还是它将获取这一步被省略掉的事实？

我的另一个客户申请条件相似——高价值房产，中等贷款额——当天晚些时候也碰到此种情形。同样地，在没有输入授权码的情况下我得以继续操作。

这里我也会去查出这个申请者的确切细节，如房产价值和贷款额度。

7.9 认证考试模拟题

在每个章节的最后，您可以试着做几道认证考试模拟题以加强对材料的记忆和理解，准备 ISTQB 高阶测试分析师的考试。

pin51.com 拼吾爱 最新最全资源分享

1. 假设一个测试分析师为银行项目工作，他正在升级 ATM 机以便让客户用所支持的信用卡取现。

对于该 ATM 机支持的信用卡而言，系统允许进行从 20～500 美元的取现。支持的信用卡有 American Express、Visa、Japan Credit Bank、Eurocard 和 MasterCard。

您正在评审您同事撰写的事件报告，报告中的重现步骤如下描述：

（1）插入 American Express 卡至 ATM 机。

（2）验证该测试账户还有 1 000 美元余额。

（3）试图从账户中取现 20 美元。

（4）错误信息显示："取现金额超出了范围"。

（5）使用其他两个具有足够额度的账户来重现该失效，即取现 20 美元。

（6）验证 ATM 机是否有足够的现金来响应请求。

（7）对于 Visa、Japan Credit Bank、Eurocard 和 MasterCard，没有出现类似的问题。

假设缺陷报告目前处于一个新的状态，意味着它需要评审。根据这个场景中提供的信息，并假定您的组织遵循 IEEE 1044 生命周期，下面的哪一项可以最确切地阐述报告的下一步将发生什么？

A 由于它没有描述有效的缺陷，则转到无效状态

B 由于它没有描述的缺陷并不重要，将转到延期状态

C 由项目利益相关者将此状态设置为"打开"以分配优先级

D 为了让测试人员检查修正情况，把状态转至"构建"状态

2. 继续前一题中描述的场景。

假设缺陷报告目前处于一个新的状态，意味着它需要评审。根据这个场景中提供的信息并假定您的组织遵循 IEEE 1044 事件分类方法，下面的 IEEE 1044 分类项中哪一个不能被分类？

A 可疑原因　　B 实际原因

C 重现性　　D 症状

3. 下面哪一项是 IEEE 829 事件报告中的部分？

A 测试项　　B 程序步骤

C 定位　　D 重现的步骤

4. 下面哪一项正确地描述了 IEEE 1044 事件管理中的步骤顺序？

A 识别、调查、处理、处置　　B 识别、处理、调查、处置

C 调查、识别、处理、处置　　D 识别、调查、移除、处置

5. 关于 IEEE 1044 事件管理规程中的研究步骤，下面哪一项的分类是您首先要做的？

A 可疑原因　　B 根源

C 解决　　D 处置

pin51.com 拼吾爱

第8章 标准和测试过程改进

经理："我们是 SEI-CMM5 的公司，所有项目遵循 CMM 开发流程。"

作者："但如果碰到你们的客户不想按照你们的流程，他们有自己的流程，或者他们根本没有流程，同时也不想建立任何流程。"

经理："但是我们的流程是 SEI- CMM 流程。"

作者："那又怎样呢？我有些客户根本对此无所谓的。是否意味着除非我的客户想遵循 CMM，不然我就不用把这些客户介绍给你们了？"

经理："那好吧，如果您的客户坚持着我们要遵循他们的流程，那我们也只能这样了。"

作者："那么如果我的客户坚持不需要任何流程呢？"

经理："那就依客户吧。"

——这是印度的一家大型 CMM5 级的软件开发公司的经理，在 2001 年与作者之间的对话，这说明了标准和过程是一个复杂的话题，人们通常把它当作市场宣传的手段而不是真正能够运用于流程改进的技术。

高级大纲的第 8 章是关于标准和测试过程改进。这一章节主要是针对测试经理的。本章没有为测试分析师设定学习的目标。但无论如何，作为一个测试分析师和测试技术分析师很可能碰到有关标准和测试过程改进的问题，因此去了解一些主要的概念对他们来说是有好处的。

此外，对于那些准备 ISTQB 高级测试分析模块认证考试的人员，关于标准的特定概念都在基础级大纲中提及到，并且也是考查的范围。因此您应该熟悉和回顾高级大纲中的第 8 章，同时为了确保掌握相关标准的学习目标，您还应该复习基础级大纲中的标准。

ISTQB 术语

能力成熟度模型：描述有效的软件开发过程关键元素的一个 5 个等级的框架，能力成熟度模型包含了在软件开发和维护中计划、工程和管理方面的最佳实践（best practice），缩写为 CMM。

pin51.com 最新最全 拼吾爱

能力成熟度模型集成：描述有效的软件产品开发和维护过程的关键元素框架，能力成熟度模型集成包含了软件开发计划、工程和管理等方面的最佳实践，是CMM指定的继承版本，缩写为CMMI。

测试成熟度模型：测试过程改进的五级阶段框架，它与能力成熟度模型（CMM）相关，后者描述了有效测试过程的关键要素，缩写为TMM。

测试成熟度模型集成：与能力成熟度模型集成（CMMI）相关的五层测试过程改进框架，描述了有效测试过程的关键因素，缩写为TMMI。

测试过程改进：用于测试过程改进的一个连续框架，描述了有效测试过程的关键要素，特别针对于系统测试和验收测试，缩写为TPI。

pin51.com 拼吾爱

第9章

测试工具和自动化

"Rex，我找到了那段控制代码，我可以用它们来使磁带库返回到第一盘带子，进行植入式自动化测试。"

"真的吗？是不是供应商告诉您的?"

"不是的，Rex，我试了很多种控制代码，终于发现了这段能加载第一盘带子的代码。"

"您不是跟我说没办法找到吗？"

"是的，我告诉过您做不了，但最后还是成功了。"

"Greg，下次您再告诉我您做不了什么事情，我肯定不相信您了。您已经告诉供应商了吗？"

"还没有，我刚才在想要给他们电话呢。"

"当然，快去给他们电话，并告诉他们得因此付钱给我们。"

——一位资深的自动化测试工程师解决了一个特别难处理的测试自动化问题后，作者与他展开的讨论。

高级大纲中的第 9 章讲解了测试工具和自动化。虽然基础级大纲也包含了这个主题，高级大纲则更深入地分析了测试工具和自动化。此外，高级大纲也阐述了基础级大纲中介绍的所有工具分类。第 9 章包括如下 3 个部分：

1. 概述。
2. 测试工具的概念。
3. 测试工具的分类。

让我们来学习每一部分并看看它们如何与测试分析相关联。

9.1 概　　述

学习目标：

能回想起该部分内容即可。

本章我们将在基础级大纲的基础上扩充一下基本工具的理念。首先是介绍一般性概

pin51.com 拼吾爱

念，接着讲解一些特定的工具。

测试人员都应该基本掌握测试工具的用途。通常，一些组织和个人给予测试工具过高的期望，并不切实际地认为合适的工具能解决所有的测试问题。

高级大纲根据工具的实际使用者，如测试经理、测试分析师和测试技术分析师来进行分类。当然，某些工具有更广泛的应用，并适用于多重角色。

9.2 测试工具的概念

学习目标:

（K2）从如下角度比较每个测试工具概念的要素和特性：风险和收益、测试工具策略、工具集成、自动化语言、测试准则、工具部署、开源工具、工具开发、工具分类。

测试工具是非常有用的，有些是必不可少的。很难想象一个超过2～3人的测试项目不使用事件跟踪系统。总地来说，测试工具可以提高测试工作的效率和精确度。不管怎样，应该认真地选择、实施并使用工具来获得收益。

人们通常认为测试自动化是测试执行的同义词，而实际上也可将测试流程的其他部分进行自动化。在人工执行一些测试任务很困难或工作量很大时，即可运用测试自动化，如回归测试和性能测试。

若想从使用测试工具中获益，您不仅需要仔细地选择和实施测试工具，还要持续地对它们进行管理。您应该有计划地为测试工具、测试脚本、测试数据和其他的测试工具项进行配置管理，并与测试、测试工具和被测系统的版本号进行绑定。您还应该为测试自动化系统创建合适的架构。作为一个顾问，我经常碰到测试团队由于在测试自动化初始阶段，在设计方面做了糟糕的决定，使他们自己备受限制。

一个好的架构能满足自动化的另一要素，即建立和维护测试库。它们拥有测试用例的数量、测试用例的命名规则、测试环境的交互等决策数据，您可以借助测试工具建立起可重复使用的测试构建块，并且还需要一个库存放这些数据。

自动化测试是测试其他程序的程序。因此，如同任何其他程序一样，由于学习的复杂性和学习所需的时间，自动化测试常常意味着您必须建立诸如它如何使用，为什么它是这样使用等类型的文档。这个文档并不需要编写得很华丽，但是大多数自动化测试系统都需要用到它。

最后，为自动化测试的扩充和维护做准备。假如不能提前考虑这些问题，特别是测试的维护性问题，这可能是测试自动化失败的一个最大原因。

后面我们将从自动化测试的业务案例着手深入地分析这些问题。记住，测试自动化应该只针对有强烈需求的业务项目，这些需求一般包括缩短测试执行周期，减少总体测试开销，或覆盖额外的质量风险。当我们在谈论这些好处时，也不要忘记去和手工测试进行比较，一项投资的回报需和其他的方案进行比较才有意义。

pin51.com 华章图书 机械工业出版社 华章公司

9.2.1　测试自动化成本

对于任何一个业务案例，我们必须考虑成本、风险和收益。首先我们从成本说起，可把成本划分为初始成本和经常性成本。初始成本包括以下几方面：

- 评估和选择合适的工具。很多公司在这一方面试图走捷径，但他们却在后来付出了更大的代价。
- 购买工具，或运用开源工具，或开发自己的工具。
- 学习和使用工具，这包含了内部知识转换和构建的成本，又包含了为测试自动化架构做设计和编写文档的成本。
- 与现有的测试流程，其他测试工具和团队进行集成。

经常性的成本包括以下几方面：

- 维护工具和测试脚本。这个问题是有关测试脚本的持续性，即在脚本进行升级前的可使用时间，它是一个很大的成本。确保设计的测试系统架构将这个成本最小化，换句话说，将测试脚本的持续性最大化。
- 持续的许可费用。
- 工具的支持费用。
- 持续的培训成本，如对新员工的培训或工具升级。
- 移植测试到新的平台。
- 扩展新特征和新功能。
- 处理由于工具可用性、限制条件和依赖性而带来的问题。
- 对测试脚本进行持续的质量改进。人们自然地也会忽略这一步。但是对于成员能力各异的团队，如果跳过这一步，意味着在工具使用和脚本发展过程中会出现参差不齐，互不兼容的问题，同时程序复用的机会将大大减少。请相信我，我的一个客户由于他们的两个自动化测试工程师在创建本质上相同的两个测试工具，使得他们的花费超过 25 万美金并错过了项目交付期。

在基础级大纲中曾推荐用一个试验性项目来引入自动化，这个主意非常好。但是，记住基于试验性项目的业务案例往往没有考虑重要的经常性成本，特别是维护的成本。

我们也可将成本划分为固定成本和可变成本。固定成本的发生与我们想进行自动化测试用例的数目无关。购买工具、培训和许可是主要的固定成本。可变成本将依赖于我们的测试次数。测试脚本开发、测试数据开发等是主要的可变成本。

9.2.2　测试自动化风险

测试自动化的风险也应该在考虑之列。除了在基础级认证大纲中涵盖的主题之外，还需要考虑以下方面：

- 现有的手工测试可能是不完备或不正确的。如果您使用这样的手工测试作为自动化测试的依据，您猜会怎样，只是加快了执行错误的事情！您需要在进行自动化之前再次检查手工测试用例、数据和脚本。因为在后期修复的代价将更大。
- 当被测试软件改变时，测试脚本、测试框架和测试数据很难维护，且需要频繁地

pin51.com 最新最全 拼吾爱

升级更新。这是一个经典的让人头痛的测试自动化例子。可维护的、健壮的、模块化的测试自动化框架设计、测试脚本和数据复用的设计，以及其他的技术将大大减少此类情况发生的可能性。如果确实发生了，那么它就是测试自动化项目的杀手，因为测试维护工作即将消耗自动化所有的可用资源，使得自动化本该带来的进步停滞不前。

- 只运行自动的、脚本化的测试用例，测试活动中缺少测试人员的直接参与，会降低缺陷发现率。自动化测试有助于建立信心，管理回归风险，以及重复测试。但是有时通过手工执行能够发现的缺陷，通过脚本却无法发现。一定程度的手工测试也是必须要保证的。

到此，您应该理解所有这些风险都是能够且是必须管理的。当自动化测试有价值时，毫无疑问，我们就应该进行测试自动化。

9.2.3 测试自动化收益

毋庸置疑，我们花费成本并承担风险的目的就是获得收益。那么测试自动化的好处是什么呢？

首先，我必须强调最明智的测试团队会投资且大量地投资于开发测试自动化的测试用例、测试数据、测试框架和其他的自动化支持项，目的就是期望年复一年地在可重复的、低维护成本的自动化测试执行上获得回报。当我在谈起“大量地投资”时，意思就是说明智的团队不会在起初的测试自动化开发方面寻求捷径，因为他们知道那将大大减少总体收益。

明智的测试团队也会有判断地选择自动化测试用例，选择那些能够从自动化中获得好处的测试用例。而那些死脑筋的人却在尝试对所有现有的手工测试用例进行自动化，最终只能以失败而告终。

一旦准备就绪，具有良好设计，经过仔细选择的自动化测试用例将可以高效运行，大大降低人工参与的工作量。由于测试自动化成本较低，测试时间短，我们可以如愿地执行测试，同时加大整体覆盖率，并对按期发布充满信心。

谈到初始投资量，您必须记住，即使不需要数年，也得在数月后，您的收益才可能等同于初始成本。我们要铭记：在多数情况下，没有捷径可寻。如果您尝试在开发上减少初始成本，那么最终自动化测试执行的收益将为 0 或小于 0。您也可以自己计算何时才能达到收支平衡。

那么，自动化测试除了能让我们节省时间、减少开销和提高覆盖率（来降低风险），还能为我们提供什么呢？

首先，我们可以更好地预测测试执行时间。如果我们开始自动化测试，让它在晚上运行，第二天早晨回来时，发现所有的测试用例都运行过了，那是一种非常好的感觉，管理层很喜欢这样。

其次，快速执行回归和验证测试的能力将创造额外的收益。因为能够控制伴随变更的风险并能更快更好地开发产品，我们不再害怕项目后期中的变更。当然，如同一把双刃剑，它也可能造成我们不够谨慎。不过只要认真地使用，它会帮助我们很好地应对紧

急情况。同时，由于某些生命周期模型在项目后期固有的频繁变更的特性，特别是敏捷和迭代模型，我们会发现将这种管理回归风险的能力，用于特定项目是很有必要的，并且不需要持续增加投入。

因为测试自动化比手工测试更具有挑战并更加受到尊重，测试人员和测试团队发现运用自动化测试是有价值的。

最后，有一些特定的测试类型不能通过手工测试进行覆盖，如性能测试和可靠性测试。有了自动化测试则可覆盖这些类型并降低其中风险。

9.2.4　测试自动化策略

我曾在很多不同的场合对客户讲过，测试工具本身并不是测试自动化策略，让我举一个非软件的例子来说明。

我现居住在德克萨斯州的中部，这里生长着巨大的、密不可见的灌木林，人们也称它们为雪松，实际上它们不是雪松，雪松是笔直生长并且是有用的树木，而这些灌木林只是长得过大。到了冬季它们传授花粉的时候，德克萨斯州中部近 400 万的人都想逃离。

我家住在大约 11 英亩（大约 4 公顷）的土地上，我们的周围都是灌木林，我妻子和我想除掉它们。我们去 Home Depot 买来链条锯子，可这并不是除掉 11 英亩灌木林的好方法。它们有的是我的两倍高。链条锯子是除掉它们的一个有用方法，但是这个策略应包括从哪里开始，哪些树木需要拖拉机，哪些可以用链条锯子来解决，如何让树木倒下，如何处理被砍掉的树木等。

下面是测试自动化的一些策略。最重要的一点是，自动化是一个长期机制，需建立一个可维护的自动化测试系统。只对那些可自动化的用例进行测试。自动化测试在执行时可自由运行并不需要人工的介入。对那些由手工完成易犯错误的任务进行自动化测试，不仅仅包含回归测试——这当然是进行自动化的高价值目标——也包括创建和载入测试数据。仅仅自动化那些测试套件，甚至只是测试套件里的测试用例，是纯粹的商业行为。您必须得清楚意识到，该被测应用程序使用期间需要重复多少次测试用例。

即使大多数的自动化测试是经过深思熟虑后的选择，也还是要考虑那些更容易使用的自动化方式。例如，如果您发现可以利用一个免费脚本来循环执行您的应用程序，以便查找可靠性问题，那么您完全可以采用这个方案。

也就是说，不论是免费的还是商业的工具，都需要仔细考虑测试工具组合。特别是当每个人都下载自己偏爱的免费工具时，工具组合很可能出现不可控的局面。在选择和评估测试工具上，也需要有一套缜密的流程，操作时不要偏离流程。

最后，要使自动化可复用和可持续。确保提供一套操作指南包括如何获得工具、如何选择要自动化的测试用例、如何去撰写可维护的脚本等。

9.2.5　测试工具集成和脚本

多种测试工具应能一起用于处理复杂的测试自动化项目。后面将通过例子来说明这一点。

很多公司使用多个测试和开发工具。他们有静态分析和单元测试工具、测试报告管

pin51.com 拼吾爱 最新最全

理工具、测试数据工具、配置管理工具、事件管理工具和图形用户界面测试执行工具。这样的组织有需要（尽管有时他们没有意识到这个需要），即把所有的测试结果集成到测试管理工具中，并增加从测试到需求（或风险）的回溯性。在这种情况下，试着去集成工具并让他们彼此交换信息。

即使您购买同一供应商的测试工具套件，也不意味着这些工具一定能够集成。这些套件应该可集成，您当然不想购买无法集成的工具。如果您不能得到完全集成的一套工具，那么就必须自行集成。将这些额外付出的努力和手工执行这些信息的成本与风险去平衡。集成开发环境有很多优势，我们测试人员也希望将来测试工具也可以这样集成。

绝大多数的自动化测试工具——至少这些执行的工具——可以运用脚本语言。一般而言，我们将在这些脚本里写入测试框架并在纯文本文件、XML 文件或数据库中使用测试数据或关键字驱动测试。这些特征将支持自动化测试工具的可维护性。

您应该记得在第 4 章中我们努力解决测试中的组合爆炸问题。由于需要付出很多，手工测试不是处理组合测试的好方法。无论如何，运用脚本语言，我们可以比手工测试创建更多组合的测试。

某些工具提供与应用的接口，比如，它们可直接通过 HTTP 和 HTTPS 服务器与网络服务器通信，而不用从浏览器进行测试输入。

脚本语言的能力差异非常大。有一些脚本语言如同一般的程序设计语言，也有一些则针对特定领域，如 TTCN-3。还有一些虽不是特定领域的，但是某些特征使得它们在某些特定的领域更加流行，如电讯/电话和嵌入式系统领域的 TCL。

不是所有的工具都需要花费金钱购买。有的工具可从网上下载，而有的可自行构建。市面上有很多种开源测试工具。对于商业软件，它们的质量参差不齐。我使用过一些很好的开源测试工具，也听说过一些非常差劲的工具。尽管不需要花钱去购买开源工具但是需要花费时间和精力去学习、使用和维护。因此需要像评估商业工具一样来评估开源工具，而不仅是运行一个演示版本。记住，通常的演示版本总是没问题的，但只能做到与基本平台兼容。

此外，出于质量的考虑，一些开源工具要求特定类型的许可证，如 Creative Commons 和 GNU Public 许可证，您也许不得不共享您所做的改进。公司的管理层想知道这样做是否真的可行。

如果找不到开源的或商业的工具，您也可以构建自己的工具。很多人都是这样做的。但这也是一个代价不菲的路子。通常，一两个人临时承担这项工作，一旦他们离开了，就存在无法使用工具的风险。要确保自主创建的工具得以文档化。

最后要注意的一点是，测试安全关键系统时需要遵循相关的标准。这在一定程度上使部分商业工具或开源工具不适用于测试此类系统。

我们来看一下集成测试系统架构的案例，如图 9-1 所示。这是一个保险公司的自动化测试系统。这个被测系统——或更确切地说，被测系统的子系统——在图中间。

在前端是 3 个主要的接口：浏览器、早先的 UNIX 绿屏应用软件和较新的 Windows 综合视图。前面的应用通过连接保险公司的网络设备，通过网络连接到后面的 iSeries 服务器。可以想象作为一家具有良好声誉的区域保险公司，iSeries 管理着一个非常大的资

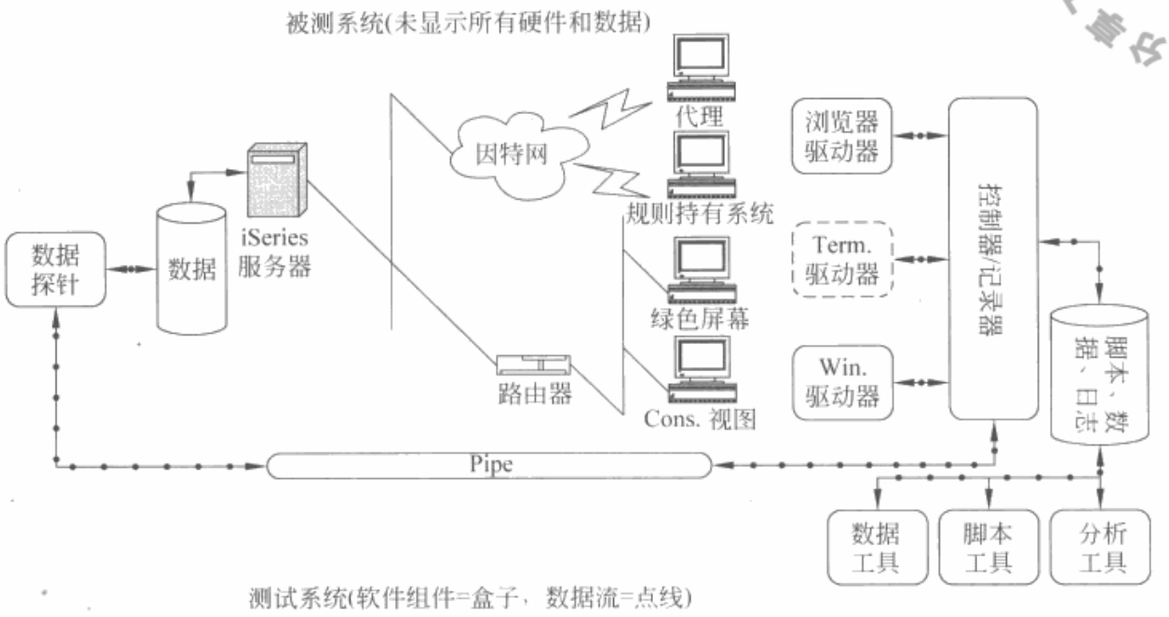


图 9-1　集成测试架构

料库，包括客户、政策、索赔记录、应付账款、应收账款等。

图的右侧是测试自动化系统的主要单元。对于 3 个接口类型中的每一个，我们需要一个驱动提交输入和获得响应。终端驱动使用点线来表示是因为起初不确定是否需要它。控制器/记录器在基于脚本库的基础上，运用驱动使测试执行同时也记录测试结果。运用工具来创建测试数据和脚本，测试日志分析也是用工具来表示。

注意到图右侧的所有这些元件都可以由单一的集成工具提供。无论如何，这是一个测试系统设计图，因此我们不关心如何实现的细节。最佳实践告诉我们首先去设计所需要的，然后寻求相应的工具，而不是让工具指使您将如何设计测试。相信我，我吃过这方面的苦头。如果让工具来驱动测试，可能导致测试了并不重要的东西。

我们再看一看图的左下方。在很多复杂的应用中，显示屏只是来显示状态如何。而真正重要的是数据转换、数据存储、数据删除和其他的数据操作。因此，要想知道测试是否通过，我们需要借助数据探针进行数据检查。

管道是一个传输结构。控制器发出的请求经由管道到达数据探针，数据探针通过管道返回结果。例如，如果启动一个特定交易，该交易会在表格中增加 100 条记录，接下来控制器执行某个程序开始交易——经由 Windows 驱动器通过 Windows 接口——然后使用探针检查新增的一百条记录。这时，在添加了 90 条记录时，显示屏的信息也可能报告显示成功。因此，我们需要一种方法来发现这种类型的缺陷，这种设计就是方法之一。

在所有可能性中，图右边实施的工具可以由一种或两种商业的或免费的工具进行集成。数据探针和管道很可能是客户定制开发的。

9.2.6　测试工具分类

贯穿本书，我们看到了不同种类的测试工具分类方法。在下一部分，我们将讲解不同的工具，由一些特定的分类来划分。那么，让我们一起来学习工具的不同分类并结束

pin51.com 最新最全 拼吾爱

这一部分。

在基础级大纲中，我们根据工具所支持的测试活动来分类。我们也可以根据测试的不同级别来分类，如单元测试工具、集成测试工具、系统测试工具。从潜在的重复使用率来看，这样的分类方式我认为并不合适。

我们可根据缺陷类型来分类。另外，也可根据这些工具所支持的测试技术来分类，比如，课程前面部分提到结对工具和分类树工具。

我们可根据工具不同的目的进行分类，如工具所支持的基本测试流程中的活动和任务。或者，我们可根据工具的应用领域分类，这适用于那些专属特定领域的测试工具，而非通用型工具。我们也可以根据工具如何被应用来分类。

9.3 测试工具类型

学习目标:

（K2）从目的、使用、优点、风险等角度对测试工具进行分类并举例。

（K2）将工具类别中的工具映射到不同的测试级别和测试类型。

与前面讨论的可选分类相对应，我们将在本章讨论根据用户的角色进行分类。在我们讨论工具和分类时，请记住一点，我是在扩展基础级大纲中介绍的工具分类，并介绍新工具类别。因此，考虑到本章节未涉及到的工具，建议您参考基础级大纲和高级大纲获取总体信息。

9.3.1 测试管理工具

测试管理工具，顾名思义，就是用来管理测试的工具。由于测试中有很多信息需要管理，包括以下信息：

- 跟踪测试工作产品到测试依据，如需求、风险等。
- 复杂环境中测试环境数据的获取。
- 跟踪并行的测试执行，包括在多个地点不同测试环境下运行的测试。
- 追溯与测试相关的各种度量，如执行测试用例的时间和花费、测试套件、回归测试集，和其他描述测试流程的度量；测试用例数，测试脚本，测试环境等；通过/失败百分比；被阻塞的测试用例数（和被阻塞条件）；不同度量的趋势，如发现缺陷/修复缺陷的比例；需求的数目和状态；测试脚本，测试设计之间的关系和可追溯性等。
- 测试管理工具通常蕴含特定的组织概念，也就是起管理的作用。它们可以作为测试用例和其他测试工作产品的存储和驱动中心。测试管理工具还能够组织测试条件和测试环境，存储并为回归套件和测试会话中的测试用例分类、处理日志和失效信息、为重启和重新初始化环境提供指导等。

测试管理工具贯穿于整个项目的生命周期，并提供给测试经理、测试分析师、测试技术分析师应用。

pin51.com 华章图书 测试人员必备

ISTQB 术语

测试管理工具：对测试过程中的测试管理和控制部分提供支持的工具。它通常有如下功能：测试件的管理、测试计划的制定、结果记录、过程跟踪、事件管理和测试报告。

9.3.2　测试执行工具

如果使用恰当的话，测试执行工具可降低成本，增加覆盖或让测试更具可重复性。测试执行工具常用于自动化回归测试。

测试执行工具通常执行被称为脚本语言的命令，是由程序语言定制编写的。工具中所用命令可以在检查图形用户界面和其他界面的过程中，精确驱动键盘输入和鼠标移动。这也有可能导致测试脚本和预期结果不稳定。

脚本可用捕捉回放工具来记录，也可构造成真正的应用程序。您可以用捕捉回放来追踪解释性的或其他无脚本测试的路径。但是实际结果和期望结果将很难维护。

测试执行工具使用比较器来比较期望结果（通常在一些前期的测试运行中捕捉）与实际结果。通常比较器会避免去比较可变的范围，如日期和时间。

如同开发程序那样，构造测试自动化脚本也是先构建功能或行动库，然后由关键字或者数据来驱动。关键字法被称作是关键字驱动或行动词驱动测试自动化。对脚本中的数据分离将克服由捕捉回放带来的可维护性问题。

糟糕的编程技能和自动化架构设计将导致测试自动化失败。这也需要仔细地管理。同时，由于测试执行脚本也是程序，我们还需要去测试它们本身。

我们更进一步看看关于关键字驱动或行为词驱动测试自动化。更确切地说，关键字将代表商业和系统之间的一些交互。例如，我们可以设置“取消订单”这个关键字，将导致整个流程中发生这样的结果。我们可以将关键字串起来创建测试步骤，这将与测试用例或整个端到端的场景相对应。关键字被当作可执行的测试脚本来实现或被看作用于任何一种被选择的工具的脚本。脚本是模块化的，应用这样的框架要求参与人员具备高超的编程技能。[1]

尽管创建测试框架的成本非常昂贵，但是这种方法提供了很多优势。首先，领域专家，基于他们对系统的理解，可以定义关键字。因此，可从用户的角度来编写测试，而不是从工具或（专业）自动化测试人员的角度。

ISTQB 术语

关键字驱动测试：一种脚本编写技术，所使用的数据文件不但包含测试数据和预期结果，还包含与被测程序相关的关键字。用于测试的控制脚本通过调用特别的辅助脚本来解释这些关键字。

1　更深入的讨论，可参考 Hans Buwalda 的 Integrated Test Design and Automation。

pin51.com 最新最全源码 拼吾爱

测试执行工具：使用自动化测试脚本执行其他软件（如捕捉/回放）的一种测试工具。

测试准则：在测试时确定预期结果与实际结果进行比较的源。一个准则可能是现有的系统（用作基准），一份用户手册，或者是个人的专业知识，但不可以是代码。

一旦框架准备就绪，领域专家就可运用关键字和他们定义的数据来自行编写测试。一旦测试用例完成，任何人都可运行它们，而并不需要特别的专长，除非测试失败。在失败的情况下，领域专家可能需要对失效进行隔离来检查是被测系统的问题还是测试框架的问题。

模块化的脚本和测试用例较易维护，是较线性的自动化方式。如果一个显示屏改变了，只要更新某一可重用脚本即可。

测试规格说明独立于具体的实施，因此可以将它们引入到框架的其他实施。事实上，多年前我们曾遇到类似情况，我们需要实施 3 个测试框架去测试一个需要在将近一打的操作系统上运行的查询工具的不同版本。和测试一样，学会如何对框架进行编程需要一段时间，但是一旦我们想出办法了，让测试运转起来就没有问题了。

测试执行工具大多为测试分析师和测试技术分析师所使用，通常他们能够在任意级别测试的执行阶段使用这些工具。尽管测试框架是由测试技术分析师创建的，但关键字测试的开发通常是由领域专家和测试分析师来完成。测试人员和领域专家可在任意级别测试的执行阶段，通过关键字测试来实现测试执行，但这更多的是应用于系统测试和验收测试。

9.3.3 调试、故障排除、缺陷散播和缺陷注入工具

调试和故障排除工具能帮助我们缩小缺陷存在范围。在一些情况下，如用户界面的缺陷，缺陷所处的位置是很明显的。但另外的一些缺陷与表面上的症状相差很远。调试工具可包括记录、跟踪和模拟环境。

调试器可以让编程人员逐行运行程序，查找意料之外的控制流或者数据流。如果编程人员预感到哪儿有缺陷或想在某一点检查变量时，调试器可在程序的运行过程中暂停程序，设置并检查程序变量。

正如在基础级大纲中提及到的，调试与测试缺陷相关但并不是测试。相似地，调试工具与测试相关但也不是测试工具。

缺陷散播和缺陷注入是不同的但相近的技术。缺陷散播使用类似于编译器的工具把缺陷植入程序中。这通常是来检查一系列测试能够发现这类缺陷的能力。当然，这个（被植入）修改的存在缺陷的版本是不应该作为产品代码被保留的！这种技术有时也称作为突变测试（Mutation testing）技术。但我还从来没有见到谁真的这样做了。

缺陷注入通常是在一个接口上注入错误的数据或事件。例如，我有一个工具可以让我随机地破坏文件内容。注意这是在第 4 章中讨论过的有关惠特克（Whittaker）的攻击技术。

调试、故障排除、缺陷散播和缺陷注入工具主要由测试技术分析师来使用。只要代

码存在，测试技术分析师可在软件生命周期中的任何时候使用这些工具。

9.3.4　静态和动态分析工具

静态分析工具对静态测试流程的某些部分进行自动化，可以在软件生命周期的任何时候使用。静态分析工具以警告的方式报告它们的发现，如代码、需求等潜在的问题。例如，代码分析工具将标志出危险的或不可靠的结构体。根据需求说明，运行拼写和语法检查器，可发现高难度的问题。

> **ISTQB 术语**
>
> **调试工具**：程序员用来复现软件失败、研究程序状态并查找相应缺陷的工具。调试器可以让程序员单步执行程序、在任何程序语句中终止程序和设置、检查程序变量。
>
> **缺陷散播工具**：在组件或系统中散播缺陷的工具（比如，故意插入一个缺陷）。
>
> **静态分析器**：执行静态分析的工具。

我们为客户使用这些工具时经常碰到的一个问题是误报数目。在这种情况下，误报是一个潜在的问题但实际上并不造成任何破坏。基于现存代码的误报数可能会很大，差不多每 5 或 10 行代码中就有一个。针对此有不同的策略，如仅对新的和变更的代码模块使用工具。幸运的是，厂商们已经意识到了这个问题并正在努力解决它。

我们来看一个关于静态分析和文本执行工具的实际使用例子。图 9-2 所示是我的同事和我为一客户构建的测试框架。这个工具使用商业的和开源的工具来提供自动化的静态分析、单元、组件和集成测试。最好从图底部的左端开始并按顺时针方向来阅读本图。我们一起看看它是如何工作的。

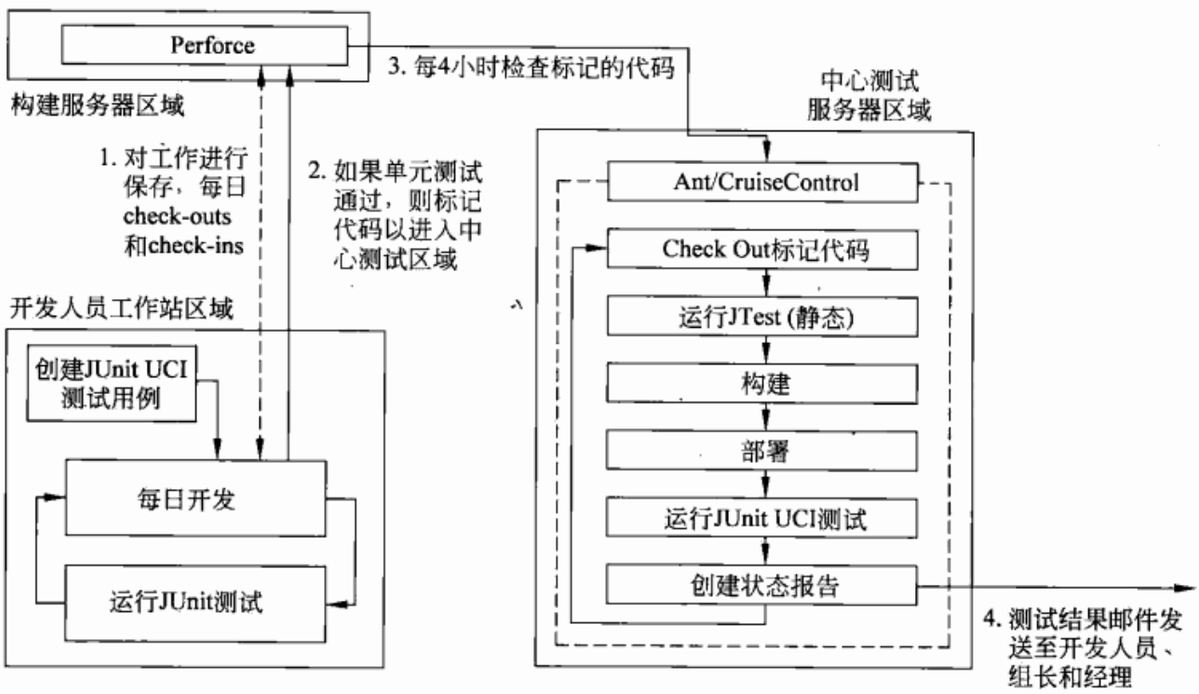


图 9-2　静态分析和单元测试流程

pin51.com 最新最全 拼吾爱

单个程序员在自己的 PC 机上工作，为他们所编写的代码来创建单元、组件和集成测试。在本机上他们用 JUnit 来运行单元测试。在每天下班时，检查他们的工作并存放到 Perforce 配置管理系统，在代码被组长审批之前，将以“当前不可构建”做标记。

一旦单元测试通过，编程人员所在小组的组长将对他们的代码和单元测试进行评审。（是的，这个评审流程没有我预想的那么正式，但这已经是我们能让他们做的全部了。）

如果评审成功，代码和测试将被检查并存放到 Perforce 中标记为“可构建”。

中心测试服务器有一个脚本每隔 4 小时运行那些被标记为“可构建”的代码。如果发现缺陷，它将重新开始一轮新的测试运行。这个测试运行包括两部分，第一部分是使用 Parasoft 公司的 JTest 工具来执行静态测试。接下来，一个完整的动态测试运行库中所有的单元、组件和集成测试。一旦测试运行完成，结果将通过邮件传送并公布在内部网上。

我认为这是一个明智的办法，也是每个开发组织应该尝试采纳的办法。注意，本案例通过合理设计，使得这些增加回归风险的行为本身——Check In 新的或更改的代码——同时可以触发减少回归风险的活动。[2]

动态分析工具提供软件运行状态的实时信息。他们可以识别出那些难以用静态方法发现的错误和难以用动态测试隔离的问题。这些工具可以用于评估指针的使用，但也许最常见的作用是检测内存泄漏。当某些编程语言（C、C++）的内存管理由程序员控制的时候，内存泄漏就很容易发生，因为他们有时会误处理。

> **ISTQB 术语**
>
> **动态分析工具**：为程序代码提供实时信息的工具。通常用于识别未定义的指针，检测指针算法和内存地址分配、使用及释放的情况以及对内存泄露进行标记。

静态和动态分析工具主要由测试技术分析师使用。当需要被分析的工作产品存在时，测试技术分析师就可以在软件生命周期的任何时候使用静态分析工具。测试技术分析师可以在任何测试级别使用动态分析工具，以作为该级别测试执行活动的一部分。

9.3.5 性能测试工具

性能测试工具主要包括两个主要组件，一个是负载生成组件，另一个是测量和分析组件。负载生成是通过脚本实现预定义的操作特征，请参见本书第 5 章中的概念。有时这些脚本被获取，尽管根据我的经验更多地是创建脚本。需要将这些脚本放入到被测系统中而不管系统接受哪一种数据。

当一个组合的脚本在性能测试工具上运行时，由一组模拟的或虚拟的用户组成的复杂组合与系统同步地进行交互。在很多情况下，许多性能测试脚本通过通信协议层如 HTTP 或 HTTPS 来再现系统和用户之间的交互作用，而非通过图形用户界面来模拟用户与系统之间的交互作用。当这些发生时，度量组件就在收集度量。如下是一些典型的

2 本案例的完整文档 Mission Made Possible 请查看 RBCS 网站 www.rbcs-us.com。

ptpress.com.cn 异步图书 新锐技术专家

度量：

- 模拟用户的数量
- 模拟用户执行的事务的数量和种类
- 由用户产生的特定事务请求的响应时间

ISTQB 术语

性能测试工具：一种支持性能测试的工具，通常有两个功能：负载生成（load generation）和测试事务（test transaction）测量。负载生成可以模拟多用户或者大量输入数据。执行时，对选定的事务的响应时间进行测量并被记录。性能测试工具通常会生成基于测试日志的报告以及负载－响应时间图表。

基于这些度量，我们可以创建不同的报告，包括负载与响应时间的图形。

性能测试是一项复杂的活动。下面是一些需要考虑的重要因素：

- 能否保证负载生成主机有足够的硬件和网络带宽？我曾看到负载生成器在运用于被测系统满载前就饱和了，这就无法完成目标。
- 您试图使用的工具的通信协议与被测系统是否相兼容？工具是否能模拟所有需要模拟的对象？
- 工具是否有足够的灵活性以保证不同的运行特征易于执行？
- 必要的监控、分析和报告机制是否可行？

为可靠性测试准备的简单的负载生成器可以自己开发，而性能测试工具通常是需要购买的或使用开源版本。最让人头痛的——也是决定是否要自己开发性能测试工具所需要的——是度量和分析部分。当我的同事和我必须去开发性能测试工具时，那通常是最难的部分。

应该牢记几点。许多有关性能的缺陷都是设计问题。我最近发现一些严重的性能问题可导致项目的失败。因此，当性能是一个关键质量风险时，应该对性能测试的关键组件进行建模和单元测试，而不是等到系统测试才进行性能测试。

我们来看一个运用集成工具进行性能测试的例子。我在本书的前面部分提到广域网中的 IVR 服务器项目。图 9-3 显示了这个系统的架构，在图的右边是广域网的 IVR 服务器，支持内容管理和客户服务的应用服务器在中间，位于左边的是客户服务代理机器。

如果在 IVR 这里按下 0，系统将通过网络的 IP 语音传输到等待状态的代理。同时，代理应答了电话，与此同时代理应该能看到，显示屏上显示这个与他对话的用户的所有信息。这个系统必须很稳定地工作，即使在服务器超负荷时也必须能工作。

因此，我们需要一种方法进行端到端的性能测试。我们可以在 IVR 这边使用一个负载生成器来发起电话呼叫。负载生成器是用脚本语言编写的，因此我们可在这些脚本中包括模拟“按下 0”。负载生成器也能通过用户中间件与正在驱动和观察显示屏的 QA 助手（这里是 SilkTest）来协调。我们实际上可以对事务进行计时，在 IVR 上按 0 来观察客户服务代理桌面的显示。我们可运行一串这类的事务，获取运行时间，并记录这些信息为以后的分析做准备。

pin51.com 最新最全源码分享下载 拼吾爱

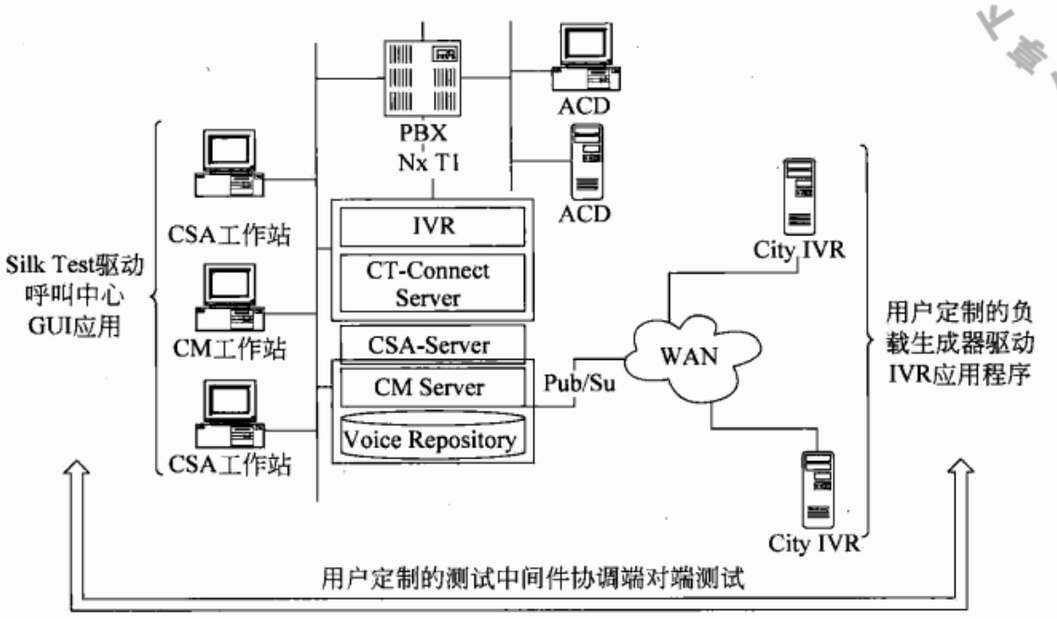


图 9-3 性能测试示例

ISTQB 术语

超链接工具：用于检查网站中超链接是否有效的工具，尚未在 ISTQB 词汇中定义。

性能测试工具主要由测试技术分析师使用。测试技术分析师可在任何一个测试级别的测试执行阶段使用这些工具，但更多的是在系统和验收测试中使用。

9.3.6 Web 测试工具

Web 工具是测试工具的另一常见类型。这些工具通常是用于扫描和检查网站中断或丢失的超链接。一些工具也提供额外的信息如链接树结构图、下载文件大小和速度、点击率和其他度量。一些工具在 HTML 上进行静态分析来检查与标准的符合度。

Web 工具主要由测试分析师和测试技术分析师使用。只要网站处于被分析状态，测试分析师和测试技术分析师可在任何一个测试级别的测试执行阶段使用这些工具。

9.3.7 模拟器和仿真器

模拟器，就如我们看的有关太空飞船的电影拍摄中运用的手法一样，提供了一种在人造环境中进行测试的方法。我们之所以这样做，是因为可能无法获取系统中的一些代码或其他部分，或者可能由于真实系统用于测试代价不菲，再或者可能因为对真实系统进行测试不安全。例如，飞机、太空飞船和核控制软件通常要在部署之前在模拟器中进行测试。我希望您会说部署构成了在实际环境下的第一轮测试。

一些模拟器可能会很复杂，有可能会注入缺陷，并产生可再生的随机伪数据流等。用模拟器来进行测试时，我的一点经验就是，不管它怎样好，在实际的硬件系统中我们总还是能发现问题。而时间问题（或时效问题）、资源限制，独立性是尤其难以准确模拟的。

仿真器是一种特殊的模拟器，由软件编写构成用于模拟硬件。我第一次碰到仿真器是在 UCLA 大学一年级的计算机科学专业的汇编语言课中。他们足够明智，并不让我们编写汇编语言在多用户 VAX 机器上运行，因为我们真的会把机器搞崩溃。他们为我们的程序提供了一个如同汇编集成的测试环境的仿真器。

使用仿真器的潜在好处在于实现更精细的测试。UCLA 大学并没有提供给我们很多的测试支持，因为它主要是设计用来阻止我们直接进入 VAX 硬件。一个设计良好的仿真器，是用来支持测试的，即能够进行跟踪、调试、记录和其他种类的“观察程序运行”活动，而这是真实的系统无法实现的。

测试分析师和测试技术分析师可根据所需要的模拟种类使用这些工具。他们可在任何一个测试级别的测试执行阶段使用这些工具，但是这些工具通常是用在每个测试级别的早期，此时被模拟或被仿真的对象尚不可用。

9.4　认证考试模拟试题

在结束每一章节时，可以通过做模拟试题来巩固所学到的知识和加深对本书的理解，同时也更好地为 ISTQB 高级测试分析师认证考试做准备。

1. 下面哪一项是自动化测试准则的例子？

 A　一个法律系统　　B　人工计算结果

 C　需求规格说明　　D　比较器

2. 假设一个测试分析师为银行项目工作，他正在升级 ATM 机以便让客户用所支持的信用卡取现。当用户第一次插入一张有效的信用卡时，系统认为该用户处于“未被验证”状态。在取现之前，用户必须输入 PIN 码来进行验证是否有效。系统应该同时支持高达 1 000 笔有效交易，同时在 2 秒钟之内完成一个用户的交易。

 下面的哪一种测试工具类型适用于上述的测试？

 A　静态分析工具　　B　性能测试工具

 C　动态分析工具　　D　Web 测试工具

3. 假设一个项目遵循某个顺序生命周期模型，下面哪个选项是关于在项目早期运用静态分析工具的例子？

 A　需求规格说明中的语法分析

 B　网络和数据库设计的性能模拟

 C　应用代码和数据库设计的安全风险分析

 D　使用 J-Unit 测试框架的自动化单元测试

pin51.com 拼吾爱 最新最全源分享社区

第10章

个人技能和团队组成

> “您输。”
>
> ——在一位女性打赌能让他说超过两个字后，美国总统Calvin Coolidge如此回复。

高级大纲第10章介绍个人技能和团队组成。本章首先分析测试人员的技能，接着扩大到内部和外部测试团队的动态性。包括讨论如何激励测试人员和测试团队，及沟通测试结果。高级大纲第10章包括如下6节内容：

1. 概述。
2. 个人技能。
3. 测试团队整体实力。
4. 组织的适合测试。
5. 激励。
6. 沟通。

这一部分的大多数内容适用于测试经理。不管怎样，让我们学习每一节，尤其是沟通这一节，并了解它们与测试分析的关联性。

10.1 概　　述

学习目标：

能回想起该部分内容即可。

本章主要讲述与管理测试团队相关的测试管理类话题。总之，主要针对的是高级测试管理认证考试的考生。由于本书讲述的是测试分析，因此我们本章中覆盖的大部分内容仅仅是回顾。

不管怎样，对于测试人员来说正确处理同事之间的关系非常重要。如何与同行进行高效沟通将会影响测试人员的成功与否。当然，您还应该不断地去提高硬技能以成为更好的测试人员。

pin51.com 新客发 新客跟单必备工具

10.2　个 人 技 能

学习目标：

能回想起该部分内容即可。

这一节的概念主要适用于测试经理。本节未对测试分析师设定学习目标。如果您在为认证考试做准备，建议阅读高级大纲中的第 10 章进行总体回顾和了解。

10.3　测试团队整体实力

学习目标：

能回想起该部分内容即可。

这一节的概念主要适用于测试经理。本节未对测试分析师设定学习目标。如果您在为认证考试做准备，建议阅读高级大纲中的第 10 章进行总体回顾和了解。

> ISTQB 术语
> 测试独立性：职责分离，有助于客观地进行测试。

10.4　组织的适合测试

学习目标：

能回想起该部分内容即可。

这一节的概念主要适用于测试经理。本节未对测试分析师设定学习目标。如果您在为认证考试做准备，建议阅读高级大纲中的第 10 章进行总体回顾和了解。

10.5　激　　励

学习目标：

能回想起该部分内容即可。

这一节的概念主要适用于测试经理。本节未对测试分析师设定学习目标。如果您在为认证考试做准备，建议阅读高级大纲中的第 10 章进行总体回顾和了解。

pin51.com 最新最全源码分享下载 拼吾爱

10.6 沟　通

学习目标：

（K2）从测试人员的角度，在考虑风险和机会的前提下，举例说明在项目中如何进行专业、客观和有效的沟通。

对于大多数测试团队，沟通分为3个级别：

- 首先，主要在内部（也和其他人）沟通相关测试文档。这包括讨论测试策略、测试计划、测试用例、测试总结报告和缺陷报告。
- 其次，沟通文档审查的反馈，主要是测试团队中内部和外部的同行沟通。包括讨论需求、功能规格说明书、用例和单元测试文档。
- 再次，沟通信息的收集与传播。这不仅包括同行沟通，也包括与经理、用户和其他的项目利益相关者进行沟通。当测试结果不利于项目成功时，这些信息有时会很敏感。

记住内部和外部沟通会影响测试分析师的专业性。

有效的沟通有助于测试分析师实现目标，相反，无效的沟通则起到阻碍作用。专业、客观和有效的沟通非常重要。不管是测试团队的内部沟通还是外部沟通，您都希望能够建立和维护测试团队的荣誉。当沟通测试结果，对文档中的问题给出反馈，或者提供一些潜在的棘手消息时，记住一定要使用合适的沟通技巧。

人们很容易在工作上感情用事，尤其是处于很紧张的测试执行阶段。记住您关注的是实现测试目标以及改进产品及其开发过程的质量。不要进行任何违背目标的沟通。

人们往往很容易与自己或者同类人沟通。换句话说，测试人员通常持着怀疑的态度简略地表达我们工作和进展的细节。当我们与同行的测试人员来讲时，这没有关系。无论如何，要记住为您的目标听众定制有效沟通。当与用户、项目团队成员、管理层、外部的测试小组和客户沟通时，必须仔细斟酌如何沟通，沟通什么，沟通是否支持您的目标。

作为测试经理，我见过粗心大意的邮件、缺陷报告或者是毫无意义的谈话，这在很大程度上毁坏了我们测试团队的名声和信誉。因此，即使您还有很多其他工作要做，还是要考虑如何沟通。[1]

图10-1是关于测试沟通的一个案例。这是一封RBCS公司发给供应商邮件的摘要部分，主要内容是关于公司对新网站的验收测试结果的（态度）。第一段主要说明这是一封经过仔细分析和推敲的邮件，而不是人们随随便便接收到的垃圾邮件。带来的信息就是“请注意阅读这封邮件”。这段还告诉读者附件中包括更多的细节。

第二段，包括列举出来的清单和结尾语，总结了需要做什么来完成验收测试并进入到部署阶段。第三段对一些特定缺陷的延期做出说明。我还想指出的一点是，我们会使

1　您可以从Judy McKay的Managing the Test People一书中读到关于测试中人员问题的精彩论述。

我花了几个小时检查现场的当前状态和验收测试情况。请查看附件中的[文档]，包括延期的缺陷报告和[测试状态]……针对 pass 2。

下面的问题必须修复，否则无法进入部署阶段：

- 消息和用户界面显示的一致性（如缺陷 85、91、92、95、97）。
- 简报链接位置与之前的决定不符（如缺陷 98）。
- 识别以及分辨内部无效链接（如缺陷 103）。

尽管有些问题在临时读者看来可能是微不足道的，但必须认清的是我们的目标客户对任何错误都会很敏感的。

但是从向前推进，及和市场并行发展方面考虑……我已经同意延期处理部分 pass1 的缺陷，不论是未通过确认测试的，还是单方面取消的，但可追溯到被宣传的产品功能。请注意这些缺陷的延期处理并不表示 RBCS 会一直接受这些缺陷的存在。

图 10-1　关于验收测试状态的邮件

用 RBCS 的法律特权来争取解决问题。最后一段巧妙地提出希望，即暗示我们对于仍然发现问题很失望。

这种类型的邮件适合于客户发给供应商，用来解释测试结果。您会发送给您的开发同事吗？也许不会吧。从这封邮件学到的重点就是邮件的每字每句都带有沟通的目的。

10.7　认证考试模拟试题

在每一章节结束时，可以通过做模拟试题来巩固所学到的知识，并加深对本书的理解，同时也将更好地为 ISTQB 高级测试分析师认证考试做准备。

1. 假设一个测试分析师为银行项目工作，他正在升级一台 ATM 机，以便客户可以用所支持的信用卡来预借现金。在项目早期，您不能获得该系统所支持的信用卡类别列表，这也是当时的需求规格说明书中没有包括的行动步骤。下面哪一项是通过邮件沟通这个问题的较好方式？

A　“除非我收到一个完整的所支持信用卡类型的清单，不然测试设计就不会有任何进展”

B　“我们什么时候才能知道支持哪种类型的信用卡？测试设计由于缺乏说明而搁置了”

C　“又是这样，需求规格说明是不完整的、模糊的”

D　不要再沟通这个问题了，就在信息系统中记录延迟吧，等着管理层追究时再来解释为什么延迟

pin51.com 拼吾爱 最新最全源分享社区

第11章

认证考试准备

> “因为我很严厉，你们大概不会喜欢我。但是你们越是恨我，学到的就越多。”
>
> ——高级军事教练对美国海军新兵训练营的新兵们如是说，
> 出自 Stanley Kubrick 的越南电影《Full Metal Jacket》。

本书第 11 章介绍的话题与 ISTQB 高级测试分析师认证考试的准备相关。本章首先讨论 ISTQB 高级测试分析师的学习目标，这也是认证考试的基础。

本章有两节：

1. 学习目标。
2. ISTQB 高级认证考试。

如果您对参加 ISTQB 高级测试分析师认证考试不感兴趣，可以跳过本章内容。

11.1 学 习 目 标

所有高级大纲的认证考试都是基于学习目标的。学习目标说明了在参加认证考试之前您应该能够做什么。每个高级认证考试有自己的一套学习目标。在每章的每一节开始，我列出了高级测试分析师认证考试的学习目标。

学习目标有 4 个递进的难度。您必须达到这些学习目标才能正确回答后面给出的认证考试问题。认证考试会覆盖比较基础的记忆和理解，还有复杂的应用和分析。例如，要想答出如何创建测试计划这样的问题，您得记住并理解 IEEE 829 测试计划模板。不同于基础级认证考试，简单记忆和理解就足够解答问题了；在高级认证考试中，必须应用或者分析您所记忆理解的事实，这样才能给出问题的正确答案。

下面来看看高级认证考试 4 个级别的学习目标。我们分别用 K1、K2、K3 和 K4 表示 4 个级别，所以复习高级大纲时，要记住这些标记。

11.1.1 级别 1：牢记（K1）

最低级别的目标，要求能够识别、牢记并且回忆起某个术语或概念。K1 的关键字有：牢记、回忆、识别和认知。本级别学习目标的考查暗含在高级别目标的问题里。

例如，能够判断出如下语句是“失效”的定义：

- “没能为终端用户或者其他利益相关者交付服务”。
- “实际的组件或系统偏离期望的交付、服务或结果”。

这就意味着，您得记住 ISTQB 高级大纲使用的 ISTQB 术语，以及 ISTQB 高级大纲参考的标准，如 ISO 9126，IEEE 829。解答较高级别 K3 和 K4 涉及的问题，需要此级别的学习。

11.1.2　级别 2：理解（K2）

本级别学习目标要求您会推论或者解释相关主题的论点，并且可以总结、区别、分类和举例。K2 应用于实际情况，所以您要能够比较不同术语的含义。您也要理解测试概念。另外，理解测试步骤，比如解释任务的顺序也是要求之一。K2 的关键字如总结、分类、比较、映射、对比、例证、解释、翻译、推断、定论和归类。

例如，您应该能够解释为什么测试应该尽早设计：

- 在缺陷造成的代价还比较小时，找出它们并移除。
- 先找出最重大的缺陷。

您也应该能够解释集成测试和系统测试的相同点和不同点：

- 相同点：测试多个组件，测试包括非功能性。
- 不同点：集成测试侧重接口和交互部分，系统测试侧重整体方面，比如，端对端处理。

以上意味着您必须理解 ISTQB 高级大纲使用到的 ISTQB 术语，以及 ISTQB 高级大纲参考的标准，如 ISO 9126，IEEE 829 的合理使用。解答较高级别 K3 和 K4 涉及的问题，需要此级别的学习。

11.1.3　级别 3：应用（K3）

本级别学习目标要求能够对概念或技术点在给定的条件下正确应用。这个级别适用于程序式知识。K3 不要求您评估软件应用或者为某个应用软件创建测试模型。提供一个模型，模型的覆盖要求，以及创建测试用例的步骤，这样的情况才属于 K3。K3 的关键字如实施、执行、使用、按照流程和应用流程。

比如，您应能做到以下两点：

- 判断出有效或无效等价划分的边界值。
- 根据一般的测试用例创建流程，从给定的状态转换图（测试用例集合）选择测试用例，以覆盖所有转换。

以上表明，您应该能够应用 ISTQB 高级大纲中的技术回答特定的认证考试题目。此级别的学习包含 K1，K2 知识。

11.1.4　级别 4：分析（K4）

本级别学习目标要求能够把一个流程或者技术的相关信息分割成几个部分，以深入理解并区分事实与推论。K4 典型的问题是，通过分析一个文档、软件或者项目现状，提出合理的做法来解决问题或者完成任务。K4 关键字有：分析、区分、选择、结构、集中、

pin51.com 最新最全源码分享 拼吾爱

归因、解析、评估、判断、监控、协调、创建、合成、生成、假设、计划、设计、构造和生产。

例如，您应能够：

- 分析项目风险，提出预防和改正的行动以减轻风险。
- 判断事件报告中哪些部分是事实性的，哪些是由结果推导出的。

这意味着您要能够分析 ISTQB 高级大纲中的技术和概念，解答特定的认证考试问题。要求此级别的学习包含 K1，K2，甚至也包括 K3。

11.1.5 学习目标级别的由来

基础级和高级大纲的学习目标分类和级别划分，参考了 Bloom 于 20 世纪 50 年代定义的学习目标分类法。如果您参加过教育培训，您可能会发现这是非常标准的教育模式。

建议从如下实际角度来看这些级别：

- K1 要求记住基本事实、技术和标准，即使您可能还不理解。
- K2 要求理解事实、技术和标准，以及它们的关联，即使您还不能把这些应用到项目中。
- K3 要求能够应用事实、技术和标准于项目中，尽管您可能还不会使它们适合您的项目，或者选择当中最合适的。
- K4 要求能够分析项目中用到的事实、技术和标准，使它们适合项目或者选择最适合的。

从中可以看出，随着学习目标的升级，要求的能力也更高。高级别更注重应用和分析。

11.2 ISTQB 高级认证考试

和基础级认证考试一样，高级认证考试也是多项选择题。多项选择题包括 3 个部分。第一部分是主干，即题目的主体。主干可能包括文字、图或表格。第二部分是干扰项。如果您没有完全理解问题覆盖到的学习目标，您就可能把干扰项当成正确答案。第三部分是正确答案。

参加过基础级认证考试，您也许认为自己同样能通过高级认证考试。这几乎是不可能的。高级认证考试主要是 K3，K4 学习目标相关的问题。换句话说，您要能够应用和分析。K1 和 K2 学习目标，作为基础级认证考试的核心，将暗含于高级目标的问题里。

例如，基础级认证考试题目一般是这样的：

下面是测试计划的主要部分，请问哪项符合 IEEE 829？

A 测试项　　B 探测影响

C 目的　　D 预期结果

正确答案是 A，而 B，C，D 都是错误选项。这个题目要求回忆 IEEE 829 模板的主要章节。只有 A 在测试计划部分。C 和 D 分别在测试规程规格说明和测试用例说明中。B 是 ISTQB 术语表中的词汇。本题只是简单的回忆。

记忆很有用，尤其在学习开始时。但仅靠回忆事实并不能让您成为一个专家。我从1970 年起是 AC/DC 乐队的主唱，深知除了记忆长长的歌词之外的能力的重要性。

高级认证考试的例题如下：

下面一段摘录于某个测试计划的测试项部分。

在系统测试期间，配置管理团队每周一早上九点前，把测试版本发给测试团队。每个测试版本要包括测试项移交报告。测试项移交报告描述该版本自动化编译的结果和冒烟测试的结果。一旦收到测试版本，如果冒烟测试是成功的，测试经理要把测试版本安装于测试实验室，然后周一早上就可以开始测试了。

如果测试团队没能收到测试版本，或者测试版本的冒烟测试不通过，或者版本没有安装，或者测试版本中没有移交报告，测试经理将立即联系配置管理团队的经理。如果问题不能在一个小时内解决，测试经理将汇报项目经理，然后有可能的话，继续上周版本的测试。如果测试版本安装失败，执行安装的测试分析师应写事件报告。

假设您是该项目的测试经理。您目前收到过两个可安装、可测试的版本。可是第三周的周一，您没有收到测试版本。下面哪个行动符合本测试计划？

A　写事件报告，描述您发现版本未安装的时间和日期

B　创建测试规程规范说明，描述如何安装测试版本

C　给配置管理经理发 SMS 信息

D　给项目经理和配置管理团队的经理发邮件

正确答案是 C。A，B 和 D 都是错误选项。A 错在不是版本没有安装，而是根本还没有到这一步。B 提到的测试规程也许对测试的安装有用，但是与本测试计划的推进没有关系。C 符合测试计划。D 不符合，原因是测试计划中提到了一小时缓冲。在这个一小时内，测试经理有可能把问题解决了，那么就不需要报告测试经理。另外，考虑到时间的重要性，电子邮件并不是好的方法。

可以看出，解答这类问题时需要分析，当然，了解 IEEE 829 的模板会有所帮助，如测试计划、事件报告、测试项转移报告和测试规程规格说明。事实上，如果您不了解标准，那么对术语可能会很迷惑。但是只是简单地了解 IEEE 829 标准并不能帮助您做出正确的解答，除非是偶然猜对的。

11.2.1　场景分析考题

很多考题进一步加强难度时通常考虑场景。在场景考题里，会描述一系列的情况，然后给出 2 个、3 个甚至更多的考题。

例如，从测试计划和错过测试发布的场景，可能延伸出另外一组问题：

假设星期一的下午您终于收到一个测试发布版本了，当测试分析师尝试安装时，安装程序中的数据库配置脚本中途退出了。数据库服务器上的错误信息是用 Cyrillic 脚本显示的，尽管所选语言是美式英语。此时，数据库表格全部崩溃。使用测试中的应用程序，都会导致各种数据库连接错误信息（至少是用美式英语显示的）。

如下是一些可能的行动：

I　通知配置管理团队的经理

pin51.com 最新最全源码资源下载 拼吾爱

II 通知项目经理
III 起草事件报告
IV 尝试重新安装
V 延迟测试
VI 继续尝试

下面哪一项描述了行动的正确顺序是合理的，同时与测试计划的意图保持一致？

A I, II, V
B V, I, IV, III, I
C VI, II, I, III, IV
D II, I, V

答案是 B。A，C 和 D 是错误的。A 错误是因为没有起草事件报告，而当安装失败时这是测试计划要求必须具有的。C 错误是因为一个有意义的测试不能在崩溃的数据库上进行，因为在通知配置管理团队经理之前先通知了项目经理，是因为在尝试重新复制失败之前起草了事件报告。D 错误是因为在通知配置管理团队经理之前先通知了项目经理，没有起草事件报告。

可以看出，对基于场景的考题，在尝试答题之前认真分析场景非常重要。如果由于匆忙地阅读使您误解了场景，可以想象得出也许您错过了部分与之相关的问题。

重新回到学习目标这个问题。我说过考题包含 K1 和 K2 学习目标的——要求牢记和理解，分别包括在更高级别的 K3 或 K4 考题中。对 K1 学习目标还增加了额外的复杂程度：它们没有被明确地定义。整个大纲，包括使用的术语和提及到的标准，都暗含覆盖了 K1 学习目标。

下面是高级大纲的一个摘录：大纲的内容，术语和所有标准的主要特征（目的）至少要记住（K1），即便没有明确地在学习目标中说明。

因此如果您正在为高级认证考试做准备，一定要多次仔细阅读高级大纲。

不仅要学习高级大纲，还要重新温习基础级大纲。下面是高级大纲的一部分摘录：

高级认证考试必须以高级大纲和基础级大纲为基础，对于一些考题的答案，可能要求运用高级大纲和基础级大纲的部分材料。高级大纲的所有部分和基础级大纲都是考查的对象。

注意上一段第 2 句话的意思是一个考题可能是高级大纲中两或三部分的交叉引用或者是高级某一部分与基础级大纲的交叉引用。因此，去做一些基础级认证考试的模拟题并重新阅读基础级大纲是为高级认证考试准备的明智之举。

11.2.2 考题的演变

高级认证考试还在不断地演变中，直到 2009 年底才能完成并统一。即使哪一天高级认证考试指南完全制定，还需要注意在高级大纲中包括一些隐晦的部分，如下：

认证考试的形式是由 ISTQB 的高级认证考试指南制定。个别委员会可根据需要采用其他的认证考试机制。

我是根据 ISTQB 高级认证考试指南初稿来编写本章的。我假设大多数的 ISTQB 国

家委员会和认证考试委员会将遵循这些。不管怎样，根据高级大纲中的这一段，对于一些委员会创建的认证考试与高级 ISTQB 高级认证考试指南相偏离也是允许的，因此在这里需要说明。您需要向本国委员会或认证考试委员会咨询并确认。他们应该在网站上公布所创建的考题是根据 ISTQB 高级指南的哪一个版本。好了，通过阅读这些，您也许会紧张。不要这样！记住认证考试是测试您对高级大纲中学习目标的掌握程度。本书包含了这些重要的特征来帮助您为认证考试做准备，如：

- 您是否做了书中的练习？如果做了，那么您掌握了最难的学习目标级别，即 K3 和 K4 目标。如果没有，则需要回头去做了。
- 您是否做了本书中的模拟题？如果做了，那么已经熟悉大纲中多数学习目标的考题。如果没有，则需要回头去做了。
- 您是否去学习在本书中出现的 ISTQB 术语定义？如果是，那么您已经熟悉了这些术语。如果没有，则需要重新学习 ISTQB 术语。
- 您是否学习了本书的每一章节和整个 ISTQB 高级大纲？如果是，您了解了 ISTQB 高级大纲中的材料。如果没有，您需要复习 ISTQB 高级大纲，并再学习那些与大纲相应的最让您迷惑的部分。

我无法保证您一定通过认证考试，但如果您认真地学习了本书、ISTQB 术语和 ISTQB 高级大纲，您将有很大几率通过高级认证考试。

祝您考试顺利！也愿您能把高级大纲中的一些理念用于您的下一个测试项目中并获得成功。

pin51.com 拼吾爱

附录A

参考文献

高级教学大纲参考标准

British Computer Society. BS 7925-2 (1998), Software Component Testing.

Institute of Electrical and Electronics Engineers. IEEE Standard 829 (1998/2007), IEEE Standard for Software Test Documentation.

Institute of Electrical and Electronics Engineers. IEEE Standard 1028 (1997), IEEE Standard for Software Reviews.

Institute of Electrical and Electronics Engineers. IEEE Standard 1044 (1993), IEEE Standard Classification for Software Anomalies.

International Standards Organization. ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality.

International Software Testing Qualifications Board. ISTQB Glossary (2007), ISTQB Glossary of terms used in Software Testing, Version 2.0.

US Federal Aviation Administration. DO-178B/ED-12B, Software Considerations in Airborne Systems and Equipment Certification.

高级教学大纲参考书籍

Beizer, Boris. *Black-Box Testing*. Wiley, 1995.

Black, Rex. *Managing the Testing Process (Second Edition)*. Wiley, 2002.

Black, Rex. *Critical Testing Processes*. Addison-Wesley, 2003.

Black, Rex. *Pragmatic Software Testing*. Wiley, 2007.

Buwalda, Hans. *Integrated Test Design and Automation*. Addison-Wesley, 2001.

Burnstein, Ilene. *Practical Software Testing*. Springer, 2003.

Copeland, Lee. *A Practitioner's Guide to Software Test Design*. Artech House, 2003.

Craig, Rick, and Stefan Jaskiel. *Systematic Software Testing*. Artech House, 2002.

Gerrard, Paul, and Neil Thompson. *Risk-Based e-Business Testing*. Artech House, 2002.

Gilb, Tom, and Dorothy Graham. *Software Inspection*. Addison-Wesley, 1993.

Graham, Dorothy, Erik van Veenendaal, Isabel Evans, and Rex Black. *Foundations of Software Testing*. Thomson Learning, 2007.

Grochmann, M. "Test Case Design Using Classification Trees." Conference Proceedings of STAR 1994.

Jorgensen, Paul. *Software Testing: A Craftsman's Approach (Second Edition)*. CRC Press, 2002.

Kaner, Cem, James Bach, and Bret Pettichord. *Lessons Learned in Software Testing*. Wiley, 2002.

Koomen, Tim, and Martin Pol. *Test Process Improvement*. Addison-Wesley, 1999.

Myers, Glenford. *The Art of Software Testing*. Wiley, 1979.

Pol, Martin, Ruud Teunissen, and Erik van Veenendaal. *Software Testing: A Guide to the T-map Approach*. Addison-Wesley, 2002.

Splaine, Steven, and Stefan Jaskiel. *The Web-Testing Handbook*. STQE Publishing, 2001.

Stamatis, D. H. *Failure Mode and Effect Analysis*. ASQ Press, 1995.

van Veenendaal, Erik, ed. *The Testing Practitioner*. UTN Publishing, 2002.

Whittaker, James. *How to Break Software*. Addison-Wesley, 2003.

Whittaker, James, and Herbert Thompson. *How to Break Software Security*. Addison-Wesley, 2004.

Wiegers, Karl. *Software Requirements (Second Edition)*. Microsoft Press, 2003.[1]

pin51.com 最新最全源码资料下载 拼吾爱

其他参考书籍

Beizer, Boris. *Software System Testing and Quality Assurance*. Van Nostrand Reinhold, 1984.

Beizer, Boris. *Software Test Techniques, 2e*. Van Nostrand Reinhold, 1990.

Jones, Capers. *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley Professional, 2000.

Koomen, Tim, et al. *TMap Next*. UTN Publishers, 2006.

McKay, Judy. *Managing the Test People*. Rocky Nook, Inc., 2007.

Nielsen, Jakob. *Usability Engineering*. Academic Press, 1993.

Tufte, Edward. *The Graphical Display of Quantitative Information, 2e*. Graphics Press, 2001.

Tufte, Edward. *Visual Explanations*. Graphics Press, 1997.

Tufte, Edward. *Envisioning Information*. Graphics Press, 1990.

White, Gregory, et al. *Security+ Certification*. Osborne, 2003.

其他参考资料

Holmes, Jeff. "Identifying Code-Inspection Improvements Using Statistical Black Belt Techniques." *Software Quality Professional*, December 2003, Volume 6, Number 1.

Black, Rex, and Greg Kubaczkowski. "Mission Made Possible." *Software Testing and Quality Engineering*, July/August 2002, Volume 4, Issue 4.

dictionary.com, for standard English words.

1. In an omission, this book is not included in the Advanced Syllabus bibliography but it is referenced in the Advanced Syllabus text. Therefore, I have included it here.

pin51.com 拼吾爱 最新最全

附录B

HELLOCARMS下一代房屋净值贷款系统

B.1 系统需求文档

本文档是RBCS公司的专有和机密材料。任何没有授权的复制、使用或者分发这些材料或者部分材料都是严厉禁止的。本文档仅提供给RBCS公司员工、授权的RBCS公司课程学员以及Rex书籍的购买者使用。

B.1.1 版本

版 本	日 期	作 者	备注	批准由/于
0.1	2007年11月1日	Rex Black	初稿	
0.2	2007年12月15日	Rex Black	第二稿	
0.5	2008年1月1日	Rex Black	第三稿	

B.1.2 术语表

术 语[1]	定 义
房屋净值	房屋的直接市场价格减去未付的住房按揭贷款余额和其他由房屋担保所产生的债务。房主可通过减少未付住房按揭贷款余额和其他由房屋担保的债务，提高房屋净值。房屋净值还可能随着资产增值而提高。 房主可根据房屋净值，通过房屋净值贷款、房屋净值信贷额度和反向住房按揭贷款的方式进行贷款
住房抵押贷款	借款人利用房产作为担保的贷款。如果借款人未能偿还贷款，借出方可以就担保房产提出合法要求，从而使这笔贷款得到保证
房屋净值贷款	一次付清的款额。在贷款启动时发放给房主以支付（给房产商），房主偿还还需要承担利息。房屋净值贷款是住房按揭贷款，由借款人的房产净值担保
房屋净值信贷额度	数量可变的金额，最高额度已预先设定。房主可视需求提取，但需支付利息。房屋净值信贷额度使得房主在需要时可取现住房按揭贷款，由借款人的房产作担保

1 术语定义来自于www.dictionary.com。

pin51.com 最新最全源码分享 拼吾爱

续表

术　语	定　义
住房按揭贷款	有法律效应的借款协议。以获得一笔钱购买房产为目的，并以房产本身作为担保
反向住房按揭贷款	另一种形式的住房按揭贷款。房主借得一笔钱，定期偿还的金额量根据房屋净值计算。比较典型的目的是使用房产净值作为资产换置基金。反向住房按揭贷款使房主在以房屋净值作为担保的条件下，可对稳定增长的住房按揭贷款住房按揭贷款取现

B.2 概　　述

在 Globobank 房屋净值贷款、房屋净值信贷额度和反向住房按揭贷款服务 HELLOCARMS 系统第一个版本发布的时候，Globobank Fairbanks 电话客服中心的客户银行经理们，就开始接受客户申请房屋净值产品（贷款、信贷额度和反向住房按揭贷款）。而在第二个版本发布时，将会允许客户通过互联网方式申请，客户对象包括 Globobank 的业务合作伙伴和顾客。

该系统的概要配置如图 B-1 所示。

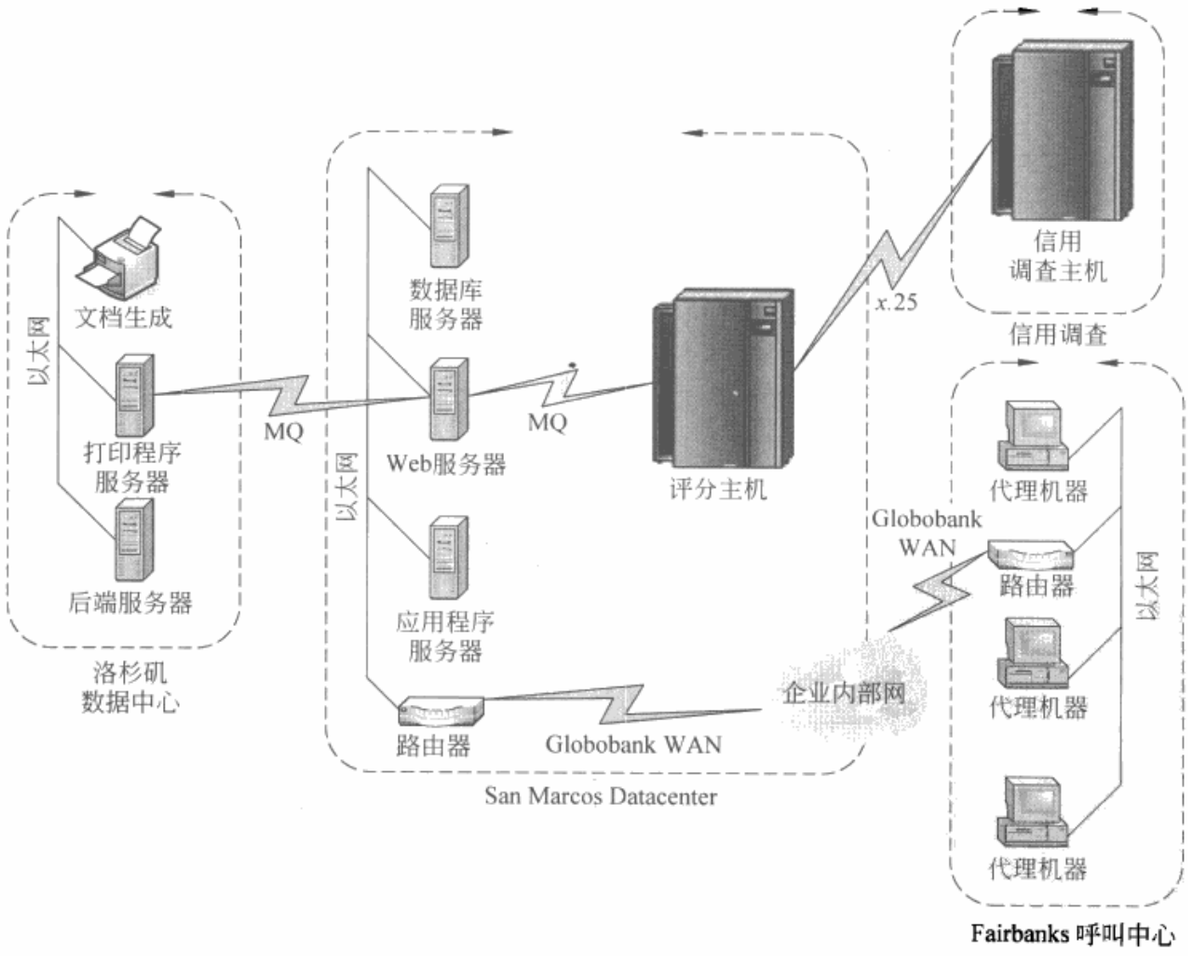


图 B-1　HELLOCARMS 系统（第一个版本）

HELLOCARMS 系统本身是一组在 Web 服务器上运行的 Java 程序和各种接口。在受

理申请过程中，数据库服务器提供存储空间，应用程序服务器则从Web服务器下载网关活动发送给客户。

B.2.1 非正式用例

下面的非正式用例适用于HELLOCARMS系统上的典型交易：

1. Globobank电话银行客户经理在Globobank呼叫中心接听到来自客户的电话。

2. 银行客户经理和客户进行电话面试，并在台式机上通过Web浏览器界面向HELLOCARMS系统输入信息。如果客户要求一大笔贷款或以担保高价值房产的方式借款，电话银行客户经理会将申请提交给更高级别的电话银行客户经理，由他来决定是否继续处理该申请。

3. 一旦电话银行客户经理收集了客户的资料，HELLOCARMS系统使用评分主机确定客户的信用价值。

4. 根据该客户的所有信息，HELLOCARMS系统显示电话银行客户经理可以提供给客户的各种房屋净值产品（如果有的话）。

5. 如果客户选择其中一款产品，电话银行客户经理将有条件地批准产品。

6. 电话面试结束后，电话银行客户经理指示HELLOCARMS系统将贷款信息传输到洛杉矶的贷款文件打印系统（LoDoPS）。

7. 假如发生下列事件，HELLOCARMS系统将收到来自LoDoPS系统的更新信息：

a. LoDoPS系统发送文件给客户。

b. Globobank贷款服务中心收到客户签字文件。

c. Globobank贷款服务中心发出符合客户选择的产品的支票或者其他材料。

一旦Globobank贷款服务中心给客户发出了所申请的资金或其他材料，HELLOCARMS就完成了对申请的处理。系统将不再跟踪该客户后续的贷款相关活动。

HELLOCARMS完成对申请的处理后，HELLOCARMS应立即将该申请及相关的所有信息归档，不管该申请最终是被银行拒绝，由客户取消，还是最后转变成有效的（active）贷款/信贷额度/反向住房按揭贷款。

B.2.2 范围

HELLOCARMS项目的范围包括：

- 从5个供应商中选择一个商业现货软件（COTS）解决方案。
- 与选定的应用程序供应商合作修改该方案，以满足Globobank的需求。
- 提供基于浏览器的前端用于贷款处理，浏览器的访问方式包括通过互联网、现有的Globobank呼叫中心、外包（非Globobank）呼叫中心、小额银行业务中心，或者经纪人。但是，HELLOCARMS的第一个版本将只提供从Globobank呼叫中心接入（确切的是Fairbanks）的方式。
- 开发连接到Globobank现有评分主机的接口，根据贷款申请和HELLOCARMS特性，为客户评分。
- 开发接口以使用Globobank现有的承销和源系统，贷款文件打印系统（LoDoPS），

pin51.com 最新最全 拼吾爱

以进行文档准备。这个接口允许 HELLOCARMS 系统，在协助客户选择产品和提供初步批准之后，转发预批准的申请（申请对象为贷款、信用额度，或反向住房按揭贷款）到 LoDoPS，并随后跟踪申请的活动状态直至服务系统。

- 接收来自 GloboRainBQW 系统的客户相关数据以生成对外邀请通知书，并通过电话、电子邮件及书面信件等方式发送给潜在的（但不是当前）Globobank 客户。

B.2.3 系统商业利益

与 HELLOCARMS 相关联的商业利益包括：

- 自动化当前的手动处理过程，并允许通过互联网和电话中心的工作人员进行贷款咨询和申请（包括从当前的呼叫中心和潜在的外包呼叫中心、小额银行业务中心和贷款经纪人）。
- 使贷款前端部分的处理时间从大约 30 分钟减少到 5 分钟。这将使 Globobank 的客户产品部门大幅增加贷款处理数量，达到其业务计划目标。
- 降低对电话银行客户经理处理贷款申请所需的技术水平的要求。因为 HELLOCARMS 将选择产品、决定申请人是否符合资格、推荐可替换的贷款产品，并提供电话银行客户经理跟进的规则。
- 通过源系统和文件预备过程，提供在线申请状态和贷款的跟踪。这将使电话银行客户经理在处理客户申请过程中，能迅速而准确地对客户咨询做出反应。
- 能够在单个环境中处理所有产品。
- 提供一致的方法，决定是否提供贷款产品给客户，若是，提供什么贷款产品，从而能够减少处理和销售错误。
- 允许基于互联网的客户（在以后的版本中）访问 Globobank 产品，选择喜爱的产品，并在几秒钟内收到初步贷款审批意见。

HELLOCARMS 系统的商业赞助商的目标是使约 85%的客户咨询获得以上这些好处，只有 15%或更少的客户咨询需提交给更高级别的电话银行客户经理做特殊处理。

B.2.4 系统功能需求

软件在特定条件下，系统能够提供满足明确要求的或者潜在需求的功能的能力。

序　号	描　述	优先级*
010–010	适用性	
010-010-010	允许电话银行客户经理接收关于房屋净值贷款、信用额度和反向住房按揭贷款的申请	1
010-010-020	提供界面和脚本两种方式，以支持呼叫中心职员完成申请贷款处理	1
010-010-030	如果客户没有提供“你是怎么知道我们的”的识别代码，在申请处理过程中，系统会通过下拉菜单搜集核心信息，菜单项是定义明确的来源分类	2
010-010-040	提供数据验证，包括使用合理的用户接口（域）控制以及后端数据验证。输入域级验证的细节有一个单独的文件描述	1

续表

序　号	描　述	优先级
010-010-050	显示已有的债务，从而可以更好地考虑选定债务的偿还能力以进行债务合并。把需要偿还的已选债务按规定传递到 LoDoPS	1
010-010-060	允许电话银行客户经理，其他 Globobank 电话销售人员和合作伙伴访问未完成或被中断的申请	2
010-010-070	询问每个申请人当前是否与 Globobank 有业务往来，例如，拥有支票或储蓄账户。发送现有 Globobank 客户关系资料到 Globobank 贷款申请数据存储中心（GLADS）	2
010-010-080	维护申请状态信息，从开始到反对、拒绝，或接受（如果接受，还包括资金交付）。	2
010-010-090	允许用户中止申请。所有屏幕上提供中止功能	3
010-010-100	允许用户在一个单独的屏幕上说明，哪些（如果有）未清偿债务，将由客户使用正在申请的资金偿还。允许用户选择特定的债务和排除特定的债务。对于尚未清偿的债务，按规定发送相关信息到 LoDoPS，说明哪些是必须使用正在申请的贷款偿还的债务	3
010-010-110	如果客户要求将（某一笔）债务还清，则从该债务比例中去除债务的月度付款	3
010-010-120	如果客户没有贷款标识符，可通过客户标识符识别已有的申请请求	4
010-010-130	如果申请的贷款额超过$ 500 000，指示电话银行客户经理将呼叫转移到高级电话银行客户经理；这类贷款需要有更高权限的管理层批准	1
010-010-140	如果申请涉及的财产价值超过$ 1 000 000，指示电话银行客户经理将呼叫转移到高级电话银行客户经理；这类申请需要有更高权限的管理层批准	2
010-010-150	对于 Globobank 运营覆盖到的所有支持的州、省和国家，提供本地和远程的电话销售支持	2
010-010-160	提供限定的合作伙伴专用的屏幕、标识、界面和品牌，以支持代理商和其他业务伙伴	2
010-010-170	支持通过互联网方式递交申请，并保证未经训练的用户也能正确地进行申请操作	3
010-010-180	提供各种功能和屏幕，支持 Globobank 的小额分行的运作	4
010-010-190	支持房屋净值申请的市场交易、销售和处理	1
010-010-200	支持房屋净值信贷额度申请的市场交易、销售和处理	2
010-010-210	支持房屋净值反向住房按揭贷款申请的市场交易、销售和处理	3
010-010-220	支持组合金融产品（如房屋净值和信用卡）申请的市场交易、销售和处理	4
010-010-230	支持原始住房按揭贷款申请的市场交易、销售和处理	5
010-010-240	支持预审批申请的市场交易、销售和处理	4
010-010-250	支持灵活的定价策略，包括建议价格、短期价格等	5
010-020	准确性	
010-020-010	确定客户具备资格申请的各项贷款、信贷额度和/或反向住房按揭贷款，并把这些选项及其计算成本和条款，提供给客户审阅。资格的确定必须与 Globobank 的信贷政策一致	1

pin51.com 最新最全源码下载 拼吾爱

续表

序　号	描　述	优先级
010-020-020	根据房产的风险、信用评分、贷款与房产价值的比例，以及收入深度，并基于评分主机的得分信息，确定客户的资格	1
010-020-030	在申请过程中，根据客户提供的申请资料评估月还款额度，将评估的月还款额度作为债务的一部分计入债务与收入比率计算，以用于信用评分	2
010-020-040	对下列房产类型增加贷款收费： ● 1.5%，对出租房产（复式房屋、公寓和度假房） ● 2.5%，对商业房产 ● 3.5%，对共管或合作房产 ● 4.5%，对未开发房产 对其他支持的房产类型，单个家庭住宅，禁止增加贷款收费	3
010-020-050	以净值记录所有政府资产换置基金的收入（例如，美国的社会保障），并在 LoDoPS 接口程序中转换为总值。[注：这是因为大部分政府资产换置收入是不收税的，但债务收入的计算中采用的是总值]	1
010-020-060	如果有的话，记录客户获得的额外收入（除工资、奖金、退休金之外）的周期（四舍五入到月）	3
010-030	互操作性	
010-030-010	如果客户在申请过程中提供“你是怎么知道我们的”的识别码，则从 GloboRainBQW 检索相关用户信息	2
010-030-020	接受联合申请（例如，合作伙伴、配偶、亲属等），评分主机将给所有申请人评分	1
010-030-030	指示评分主机删除联合申请者信用报告中重复的信用信息	2
010-030-040	允许用户在一个单独的显示屏幕上说明，哪些（如果有）是现有债务，应由客户使用正在申请的资金偿还。允许用户选择特定的债务和排除特定的债务。对于必须偿还的债务，发送一项规定条款到 LoDoPS，说明哪些是使用借入资金偿还的债务	1
010-030-060	如果评分主机没有显示取消抵押品赎回权或破产的解除日期，但是客户表明抵押品赎回权已经恢复或破产已经解除，则需继续处理有关申请，并指示电话银行客户经理要求申请人提供解除的书面证明，发送到 LoDoPS	3
010-030-070	允许用户在一个单独的显示屏幕上说明，哪些（如果有）是现有债务，应由客户使用正在申请的资金偿还。允许用户选择特定的债务和排除特定的债务。对于必须偿还的债务，发送一项条款到 LoDoPS，说明哪些是使用借入资金偿还的债务	3
010-030-080	以净值记录所有政府资产换置基金的收入（例如，美国的社会保障），并在 LoDoPS 接口程序中转换为总值。[注：这是因为大部分政府资产换置收入是不收税的，但债务收入的计算中采用的是总值]	1
010-030-090	传递申请信息到评分主机	1
010-030-100	从评分主机接收评分结果和决策信息	1
010-030-110	如果评分主机出故障了，对申请信息请求进行排队处理	2
010-030-120	一旦已验证的贷款发送到 LoDoPS，则启动源处理	2
010-030-130	传递所有被拒绝的申请到 LoDoPS	2
010-030-140	从 LoDoPS 接收申请状态的反馈信息	2

续表

序　号	描　述	优先级*
010-030-145	接收 LoDoPS 中贷款信息的变更（如贷款数量，比例等）	2
010-030-150	支持计算机电话集成，为本地电话促销活动和品牌业务伙伴提供个性化的市场和销售支持	4
010-040	安全性	
010-040-010	支持商定的安全要求（加密、防火墙等）	2
010-040-020	为每个申请跟踪“创建者”和“最后改动者”信息	1
010-040-030	允许外包电话推销员查看信贷级别，但不能查看申请人的实际信用评分	2
010-040-040	支持通过互联网提交申请，提供防止无意和有意安全攻击的手段	2
010-040-050	允许互联网用户浏览潜在贷款，而无需这些用户透露他们的姓名，政府的识别号码等个人信息，直至在申请过程中的最新可行点	4
010-040-060	对所有金融申请的处理，支持欺诈侦测	1
010-050	一致性（功能标准/法律/规则）	
	[在随后版本中待定]	

*优先级定义：

1 非常高；

2 高；

3 中等；

4 低；

5 很低。

B.2.5 系统可靠性需求

该系统在特定条件下能够保持指定级别性能的能力。

序　号	描　述	优先级
020-010	成熟性	
	[下个版本中待定]	
020-020	容错性	
	[下个版本中待定]	
020-030	可恢复性	
	[下个版本中待定]	
020-040	一致性（可靠性标准/法律/规定）	
	[下个版本中待定]	

B.2.6 系统易用性需求

系统在特定条件下，能够被用户和呼叫中心人员理解、学习、使用并认为有吸引力的能力。

pin51.com 最新最全源代码 拼吾爱

序　号	描　述	优先级
030-010	易理解性	
030-010-010	支持通过互联网提交申请，并且能使未经训练的用户进入申请程序	2
	[下个版本中将有更多内容]	
030-020	易学习性	
	[下个版本中待定]	
030-030	易操作性	
030-030-010	提供完全用户定制化方式，适用于用户界面和用户为业务伙伴提供的文档，包括销售和市场信息的私人品牌，以及所有结账凭证	3
	[下个版本中将有更多内容]	
030-040	吸引力	
	[下个版本中待定]	
030-050	一致性（可用性标准）	
030-050-010	遵从当地残疾人访问法规	5

B.2.7 系统效率需求

系统在特定条件下使用特定数量资源条件下能够提供适当性能的能力。

序　号	描　述	优先级
040-010	时间行为	
040-010-010	用户可享受 1 秒或更少的响应时间进行屏幕到屏幕切换。这一要求的测量应当从请求进入申请系统直到屏幕响应离开应用程序服务器，也就是说，不包括网络传输的延迟。	2
040-010-020	5 分钟之内对递交的申请做出批准或拒绝	2
040-010-030	在一个小时之内发起贷款，包括支付资金	3
	[下个版本中将有更多内容]	
040-020	资源消耗	
040-020-010	每小时处理最多 2 000 个申请	2
040-020-020	每小时处理最多 4 000 个申请	3
040-020-030	可应付最多 4 000 个并发申请提交	4
040-020-040	营运第一年支持总量达 120 万的经批准的申请	2
040-020-050	营运第一年支持总量达 720 万的申请	2
040-020-060	营运第一年支持总量达 240 万有条件地批准的申请	2
	[下个版本中将有更多内容]	
040-030	一致性（性能标准）	
	[下个版本中待定]	

B.2.8 系统可维护性需求

系统能够被修正的能力。修正包括改正、改进或者改编软件在环境和功能性需求方

面的变更。

序　号	描　述	优先级
050-010	易分析性	
	[下个版本中待定]	
050-020	易改变性	
	[下个版本中待定]	
040-030	一致性（性能标准）	
	[下个版本中待定]	

B.2.9 系统可移植性需求

系统能够从一个环境移植到另外一个环境的能力。

序　号	描　述	优先级
060-010	适应性	
	[下个版本中待定]	
060-020	易安装性	
	[下个版本中待定]	
060-030	可共存性	
060-030-010	不应以任何非指定的方式与 Globobank 呼叫中心或数据中心的任何其他申请交互	1
	[下个版本中将有更多内容]	
060-040	可替换性	
	无	
060-050	一致性	
	[下个版本中待定]	

B.3 致　谢

本文档来自于一个实际项目。RBCS 在此衷心感谢我们的客户同意 RBCS 改编和发表该项目多个文档的匿名部分。

pin51.com 最新最全资源分享下载 拼吾爱

附录C

问题答案

第1章

1. A、D　　2. C

第2章

1. B、D　　2. A　　3. B　　4. D

第3章

1. D　　2. B　　3. A

第4章

1. D	2. C	3. B	4. A	5. C	6. D
7. C	8. A	9. A	10. D	11. B	12. A
13. B	14. C	15. B	16. D	17. B	18. A
19. C	20. A	21. C	22. B	23. C	24. B
25. B					

第5章

1. D　　2. A　　3. B　　4. C　　5. C

第6章

1. D　　2. B　　3. C　　4. A

第7章

1. C　　2. B　　3. D　　4. A　　5. B

第9章

1. A　　2. B　　3. A

第10章

1. B